

agent2d

スタートアップマニュアル

秋葉原プログラミング教室

サッカー部編

2019/11/23

目次

内容

目次.....	1
1. セットアップ	4
2. agent2d の使い方	4
(1) 参考資料.....	4
RoboCup サッカー2D シミュレーションリーグ解説：仕組みと環境構築.....	4
RoboCup サッカー2D シミュレーションリーグ解説：サンプルエージェントを使った チーム開発	4
ロボカップサッカーシミュレーション 2D リーグ必勝ガイド.....	4
RoboCup サッカー2D シミュレーション講習会@秋キャンプ 2011	4
RoboCup サッカー2D シミュレーション講習会@ 秋キャンプ 2013.....	4
(2) librcsc(ライブラリ).....	5
①概要.....	5
②使い方	6
③agent2d のファイルを修正する	7
(3) agent2d のフォルダ構造	9
①コンパイル前	9
②コンパイル後	15
(4) コンパイル	26
①コンパイル（ビルドの過程）	26
②Make.....	26
③GNU Autotool	26
④agent2d でファイルを追加した時	28
⑤agent2d のライブラリをプログラムに同梱する方法	28
(5) src フォルダの中の「start.sh」の内容.....	30
①LIBPATH の書き込み（12行～20行）	30
②実行ファイル、conf ファイル、port 番号などを変数に入れる（24行～53行） ...	30
③ヘルプの内容表示（55行～80行）	31
④引数の処理（82行～262行）	33
⑤その他の処理（264行～）	34
⑥start.sh を実行した後にコンソールに表示される内容	34
(6) 代表的なオプション.....	41
①リモート接続（-h, --host HOST）	41

②オフラインログ (--offline-logging)	41
3. フォーマーションの変更	42
(1) フォルダ構造	42
(2) fedit2 の使い方	43
検討事項 (2019/4/7)	43
4. Librcsc の解説	45
(1) SoccerAgent クラス	45
(2) PlayerAgent クラス	45
(3) PlayerObject クラス	45
(4) WorldModel クラス	45
(5) BasicSocket クラス	45
(6) AbstractPlayerObject クラス	46
(7) InterceptTable クラス	46
(8) SoccerIntention クラス	46
(9) BodySmartkick クラス	46
(10) CoachAgent クラス	46
(11) AbstractAction クラス	46
(12) BodyAction クラス	48
(13) SoccerBehavior クラス	49
(14) BasicClient クラス	49
5. agent2d の解説	50
(1) プログラムの開始までの流れ	50
①トップフォルダの「start.sh」	50
②src フォルダの「start.sh」	50
③src フォルダの「main_player.cpp」	50
④世界モデル更新	51
⑤ログを見ながら選手の行動を分析する	51
(2) 全体のコントロール (src/sample_player.cpp)	52
(3) ボールキックの行動評価関数 (sample_field_evaluator.cpp)	55
(4) ポジショニング動作 (bhv_basic_move.cpp)	56
(5) プレイヤーの役割 (soccer_role.cpp)	57
(6) 戦略 (strategy.cpp)	59
(7) コミュニケーションルール (sample_communication.cpp)	59
(8) アクションチェイン	60
①chain_action /action_generator.h	61
②chain_action /action_chain_graph.cpp	64

③chain_action /bhv_chain_action.cpp	64
(9) シュート	65
①chain_action/shoot_generator.cpp	65
②chain_action/bhv_strict_check_shoot.cpp	65
(1 0) パス	65
①chain_action/self_pass_generator.cpp	65
②chain_action/strict_check_pass_generator.cpp	66
③chain_action/bhv_pass_kick_find_receiver.cp	67
(1 1) ドリブル	67
参考情報	68
(1) Agent2d 関連	68
(2) 論文等	68
RoboCup サッカーにおけるニューラルネットワークによる評価関数モデリング	68
RoboCup サッカー2D シミュレーションに対するゲームログからの評価関数の設定	68
RoboCup サッカーシミュレーションリーグ 2D における 局面評価関数の学習	68
RoboCup サッカーにおけるセットプレイのマーク割当手法の性能調査	68
エージェント配置問題における三角形分割を利用した近似モデル	68
自己組織化マップを用いた教師あり学習 によるエージェント協調動作獲得手法	69
(3) 世界大会出場チームのアルゴリズム説明 (Team Description Paper)	70
(4) C、C++関連	70
(5) Git、Github	71
①参考情報	71
②プログラムのバックアップ方法	71
1. ファイルをステージに追加	72
2. ファイルをコミット	72
3. ローカルリポジトリを GitHub (リモートリポジトリ) と同期	72
4. GitHub (リモートリポジトリとローカルリポジトリを同期	72
5. 以前のファイルへの戻し方	72
6. 直前のコミットのやり直し方	74
(6) virtualbox の容量が足りなくなった時	75
①Dropbox の容量削減	76
②ディスク容量の追加	76
(7) 指数表記の早見表	77
(8) RoboCup シミュレーションリーグ秋 (冬) キャンプ 2019	77

1. セットアップ

こちらのページを参考にセットアップして下さい。

<https://github.com/mmochizuki/robocup2d/wiki/%E3%82%BB%E3%83%83%E3%83%88%E3%82%A2%E3%83%83%E3%83%97>

2. agent2d の使い方

(1) 参考資料

RoboCup サッカー2D シミュレーションリーグ解説：仕組みと環境構築

https://www.jstage.jst.go.jp/article/jsoft/23/5/23_714/pdf/-char/ja

RoboCup サッカー2D シミュレーションリーグ解説：サンプルエージェントを使ったチーム開発

https://www.jstage.jst.go.jp/article/jsoft/23/6/23_838/pdf

ロボカップサッカーシミュレーション 2D リーグ必勝ガイド

<https://jaist.dl.osdn.jp/rctools/46021/RoboCup2DGuideBook-1.0.pdf>

RoboCup サッカー2D シミュレーション講習会@秋キャンプ 2011

下記のページの「講習会」と記載してある部分の「occersim2d-slide.pdf」というファイルをダウンロードして下さい。

<http://rc-oz.osdn.jp/pukiwiki/index.php?cmd=read&page=Event%2F2011%2FCamp&word=2011>

RoboCup サッカー2D シミュレーション講習会@ 秋キャンプ 2013

<http://rc-oz.osdn.jp/pukiwiki/index.php?cmd=read&page=Event%2F2013%2FCamp>

下記のページの「講習会」と記載してある部分の「camp13.pdf」というファイルをダウンロードして下さい。

サッカーシミュレーションリーグ - 情報処理学会電子図書館

https://ipsj.ixsq.nii.ac.jp/ej/index.php?action=pages_view_main&active_action=repository_action_common_download&item_id=70557&item_no=1&attribute_id=1&file_no=1&page_id=13&block_id=8

(2) librcsc(ライブラリ)

①概要

agent2d は librcsc というライブラリを使ってプログラムされています。librcsc は boost を使用して作られています。librcsc については、ロボカップサッカーシミュレーション 2D リーグ必勝ガイドの「第3章チーム開発 (p 39~)」をご覧ください。

<https://jaist.dl.osdn.jp/rctools/46021/RoboCup2DGuideBook-1.0.pdf>

また、必勝ガイドの p 46 から解説されている「Doxygen によるリファレンス生成」によってリファレンスを作成しました。下記のページの「librcsc_index.zip」をダウンロードして、フォルダの中にある「index.html」ファイルをクリックすると、ライブラリの構造が分かります。必勝ガイドの内容から少し変わっていますので、プログラムを修正する際はこちらをご覧ください。

<https://github.com/mmochizuki/robocup2d>

ldd コマンドを使って agent2d の実行ファイルである、「sample_player」をチェックしたところ、下記のようにライブラリを使っていることが分かりました。

```
mm@mm-VirtualBox:~/t1/src$ ldd sample_player
linux-vdso.so.1 => (0x00007ffe4d923000)
librcsc_agent.so.7 => /home/mm/local/lib/librcsc_agent.so.7 (0x00007eff742ef000)
librcsc_ann.so.1 => /home/mm/local/lib/librcsc_ann.so.1 (0x00007eff740e5000)
librcsc_net.so.0 => /home/mm/local/lib/librcsc_net.so.0 (0x00007eff73ee0000)
librcsc_time.so.0 => /home/mm/local/lib/librcsc_time.so.0 (0x00007eff73cdd000)
librcsc_param.so.3 => /home/mm/local/lib/librcsc_param.so.3 (0x00007eff73ac6000)
librcsc_gz.so.0 => /home/mm/local/lib/librcsc_gz.so.0 (0x00007eff738ba000)
librcsc_rcg.so.5 => /home/mm/local/lib/librcsc_rcg.so.5 (0x00007eff7367f000)
librcsc_geom.so.7 => /home/mm/local/lib/librcsc_geom.so.7 (0x00007eff73444000)
libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007eff730c2000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007eff72db9000)
libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007eff72ba3000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007eff727d9000)
libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x0000
```

②使い方

次に必勝ガイド 54 ページの `AngleDeg` という計算クラスライブラリを使って、角度の計算をしたプログラム (`AngleDeg.cpp`) を下記に記載します。ライブラリをインクルードするパスを絶対パスで記載しています。

```
-----  
#include <iostream>  
#include </home/mm/local/include/rcsc/geom/angle_deg.h>  
  
int main(int argc, char *argv[]){  
  
    rcsc::AngleDeg angle1( 100.0 );  
    rcsc::AngleDeg angle2( 30.0 );  
    rcsc::AngleDeg answer( 0 );  
  
    answer = angle1 + angle2;  
    std::cout <<"angle1:" <<angle1 << std::endl;  
    std::cout <<"angle2:" <<angle2 << std::endl;  
    std::cout <<"angle1+angle2=" <<answer << std::endl;  
  
    return 0;  
}
```

このプログラムをコンパイルして実行したところ、次のような結果となりました。

```
-----  
angle1:100  
angle2:30  
angle1+angle2=130  
-----
```

上記のプログラムとコンパイルするための `Makefile` を、「`agent2d_test`」というフォルダに入れておきますので興味のある方はコンパイルしてみてください。最初にライブラリをインクルードするパスを絶対パスを修正し、「`make -f Makefile.AngleDeg`」とコマンドを打つとコンパイルされ、「`./AngleDeg`」で実行できます。

<https://github.com/mmochizuki/robocup2d>

③agent2d のファイルを修正する

必勝ガイドの 72 ページに下記のように「Body_GoToPoint」関数の使い方が記載されています。

```
-----  
#include <rcsc/action/body_go_to_point.h>  
Body_GoToPoint( const Vector2D & target_point,  
const double & dist_thr,  
const double & dash_power,  
const int cycle = 100,  
const bool back_mode = false,  
const bool save_recovery = true,  
const double & dir_thr = 12.0 );
```

target point の位置へ、dash power のパワーで距離が dist thr 以下になるまで移動する。cycle サイクル後にちょうど目標位置へ到達するように、ダッシュパワーは自動調整される。back mode = true であれば、後方ダッシュを実行する。save recovery = true であれば、recover の値を減らさないようにダッシュパワーが自動調整される。移動中の target point の方向と体の方向との誤差が dir thr は以下である。

「Body_GoToPoint」関数は、src フォルダの中にある「sample_player.cpp」の 728 行以降などで使われています。

(修正前)

```
Body_GoToPoint( heard_pos,  
                0.5,  
                ServerParam::i().maxDashPower(),  
                ).execute( this );
```

修正前は、「const Vector2D & target_point, const double & dist_thr, const double & dash_power」に対応する引数しか記載されておらず、残りはデフォルト引数でしたので、下記のように 1 行追加して、「const int cycle = 100、const bool back_mode = false、const bool save_recovery = true」の引数を下記のように修正しました。

(修正後)

```
Body_GoToPoint( heard_pos,  
                0.5,  
                ServerParam::i().maxDashPower(),  
                50, true, false // 1 行追加  
                ).execute( this );
```

修正後にトップフォルダで「./configure」、「make」コマンドを実行すると、上記の内容が反映された実行ファイルができます。ちなみに、C++では「save_recovery = false」というように直接引数を指定して修正することができないため、修正したい引数の位置まで順番に値を入れていく必要があります。

(3) agent2d のフォルダ構造

agent2d は C++ で書かれているため、実行するにはコンパイルが必要となります。コンパイル前の agent2d は、ファイルとフォルダを合わせて 234 個、容量は 1.9MB です。コンパイル後の agent2d は、ファイルとフォルダを合計で 332 個、容量は 86.8MB まで増えています。プログラムを修正する時には、自分でコンパイルが必要となるため、フォルダ構造とコンパイルの過程について説明します。

①コンパイル前

コンパイル前の agent2d は、ファイルとフォルダを合わせると 234 個、容量は 1.9MB です。src フォルダの中にある、「bhv_basic_move.cpp」、「bhv_basic_tackle.cpp」、「bhv_normal_dribble.cpp」などのファイルの中に、librcsc のライブラリを使って一つ一つの動作を作り込んでいます。これらのファイルのパラメーターを変化させれば、動きを変化させることができます。

— AUTHORS

— COPYING

— ChangeLog

— INSTALL

— Makefile.am

— Makefile.in

— NEWS

— NEWS.en

— README

— aclocal.m4

— bootstrap

— config

| — depcomp

| — install.sh

| — missing

— config.h.in

— configure

— configure.ac

— m4

| — ax_boost_base.m4

— src

| — Makefile.am

- |—— Makefile.in
- |—— bhv_basic_move.cpp
- |—— bhv_basic_move.h
- |—— bhv_basic_offensive_kick.cpp
- |—— bhv_basic_offensive_kick.h
- |—— bhv_basic_tackle.cpp
- |—— bhv_basic_tackle.h
- |—— bhv_custom_before_kick_off.cpp
- |—— bhv_custom_before_kick_off.h
- |—— bhv_go_to_static_ball.cpp
- |—— bhv_go_to_static_ball.h
- |—— bhv_goalie_basic_move.cpp
- |—— bhv_goalie_basic_move.h
- |—— bhv_goalie_chase_ball.cpp
- |—— bhv_goalie_chase_ball.h
- |—— bhv_goalie_free_kick.cpp
- |—— bhv_goalie_free_kick.h
- |—— bhv_penalty_kick.cpp
- |—— bhv_penalty_kick.h
- |—— bhv_prepare_set_play_kick.cpp
- |—— bhv_prepare_set_play_kick.h
- |—— bhv_set_play.cpp
- |—— bhv_set_play.h
- |—— bhv_set_play_free_kick.cpp
- |—— bhv_set_play_free_kick.h
- |—— bhv_set_play_goal_kick.cpp
- |—— bhv_set_play_goal_kick.h
- |—— bhv_set_play_indirect_free_kick.cpp
- |—— bhv_set_play_indirect_free_kick.h
- |—— bhv_set_play_kick_in.cpp
- |—— bhv_set_play_kick_in.h
- |—— bhv_set_play_kick_off.cpp
- |—— bhv_set_play_kick_off.h
- |—— bhv_their_goal_kick_move.cpp
- |—— bhv_their_goal_kick_move.h
- |—— chain_action

- | |—— actgen_action_chain_length_filter.h
- | |—— actgen_cross.cpp
- | |—— actgen_cross.h
- | |—— actgen_direct_pass.cpp
- | |—— actgen_direct_pass.h
- | |—— actgen_self_pass.cpp
- | |—— actgen_self_pass.h
- | |—— actgen_shoot.cpp
- | |—— actgen_shoot.h
- | |—— actgen_short_dribble.cpp
- | |—— actgen_short_dribble.h
- | |—— actgen_simple_dribble.cpp
- | |—— actgen_simple_dribble.h
- | |—— actgen_strict_check_pass.cpp
- | |—— actgen_strict_check_pass.h
- | |—— action_chain_graph.cpp
- | |—— action_chain_graph.h
- | |—— action_chain_holder.cpp
- | |—— action_chain_holder.h
- | |—— action_generator.h
- | |—— action_state_pair.h
- | |—— bhv_chain_action.cpp
- | |—— bhv_chain_action.h
- | |—— bhv_normal_dribble.cpp
- | |—— bhv_normal_dribble.h
- | |—— bhv_pass_kick_find_receiver.cpp
- | |—— bhv_pass_kick_find_receiver.h
- | |—— bhv_strict_check_shoot.cpp
- | |—— bhv_strict_check_shoot.h
- | |—— body_force_shoot.cpp
- | |—— body_force_shoot.h
- | |—— clear_ball.cpp
- | |—— clear_ball.h
- | |—— clear_generator.cpp
- | |—— clear_generator.h
- | |—— cooperative_action.cpp

- | |—— cooperative_action.h
- | |—— cross_generator.cpp
- | |—— cross_generator.h
- | |—— dribble.cpp
- | |—— dribble.h
- | |—— field_analyzer.cpp
- | |—— field_analyzer.h
- | |—— field_evaluator.h
- | |—— hold_ball.cpp
- | |—— hold_ball.h
- | |—— neck_turn_to_receiver.cpp
- | |—— neck_turn_to_receiver.h
- | |—— pass.cpp
- | |—— pass.h
- | |—— pass_checker.h
- | |—— predict_ball_object.h
- | |—— predict_player_object.h
- | |—— predict_state.cpp
- | |—— predict_state.h
- | |—— self_pass_generator.cpp
- | |—— self_pass_generator.h
- | |—— shoot.cpp
- | |—— shoot.h
- | |—— shoot_generator.cpp
- | |—— shoot_generator.h
- | |—— short_dribble_generator.cpp
- | |—— short_dribble_generator.h
- | |—— simple_pass_checker.cpp
- | |—— simple_pass_checker.h
- | |—— strict_check_pass_generator.cpp
- | |—— strict_check_pass_generator.h
- | |—— tackle_generator.cpp
- | |—— tackle_generator.h
- |—— coach.conf
- |—— communication.h
- |—— formations-dt

- | |—— before-kick-off.conf
- | |—— defense-formation.conf
- | |—— goal-kick-opp.conf
- | |—— goal-kick-our.conf
- | |—— goalie-catch-opp.conf
- | |—— goalie-catch-our.conf
- | |—— goalie-formation.conf
- | |—— indirect-freekick-opp-formation.conf
- | |—— indirect-freekick-our-formation.conf
- | |—— kickin-our-formation.conf
- | |—— normal-formation.conf
- | |—— offense-formation.conf
- | |—— setplay-opp-formation.conf
- | |—— setplay-our-formation.conf
- |—— formations-keeper
 - | |—— before-kick-off.conf
 - | |—— defense-formation.conf
 - | |—— goal-kick-opp.conf
 - | |—— goal-kick-our.conf
 - | |—— goalie-catch-opp.conf
 - | |—— goalie-catch-our.conf
 - | |—— goalie-formation.conf
 - | |—— indirect-freekick-opp-formation.conf
 - | |—— indirect-freekick-our-formation.conf
 - | |—— kickin-our-formation.conf
 - | |—— normal-formation.conf
 - | |—— offense-formation.conf
 - | |—— setplay-opp-formation.conf
 - | |—— setplay-our-formation.conf
- |—— formations-taker
 - | |—— before-kick-off.conf
 - | |—— defense-formation.conf
 - | |—— goal-kick-opp.conf
 - | |—— goal-kick-our.conf
 - | |—— goalie-catch-opp.conf
 - | |—— goalie-catch-our.conf

- | |—— goalie-formation.conf
- | |—— indirect-freekick-opp-formation.conf
- | |—— indirect-freekick-our-formation.conf
- | |—— kickin-our-formation.conf
- | |—— normal-formation.conf
- | |—— offense-formation.conf
- | |—— setplay-opp-formation.conf
- | |—— setplay-our-formation.conf
- |—— intention_receive.cpp
- |—— intention_receive.h
- |—— intention_wait_after_set_play_kick.cpp
- |—— intention_wait_after_set_play_kick.h
- |—— keepaway.sh.in
- |—— keepaway_communication.cpp
- |—— keepaway_communication.h
- |—— main_coach.cpp
- |—— main_player.cpp
- |—— main_trainer.cpp
- |—— neck_default_intercept_neck.cpp
- |—— neck_default_intercept_neck.h
- |—— neck_goalie_turn_neck.cpp
- |—— neck_goalie_turn_neck.h
- |—— neck_offensive_intercept_neck.cpp
- |—— neck_offensive_intercept_neck.h
- |—— player.conf
- |—— role_center_back.cpp
- |—— role_center_back.h
- |—— role_center_forward.cpp
- |—— role_center_forward.h
- |—— role_defensive_half.cpp
- |—— role_defensive_half.h
- |—— role_goalie.cpp
- |—— role_goalie.h
- |—— role_keepaway_keeper.cpp
- |—— role_keepaway_keeper.h
- |—— role_keepaway_taker.cpp

- |—— role_keepaway_taker.h
- |—— role_offensive_half.cpp
- |—— role_offensive_half.h
- |—— role_sample.cpp
- |—— role_sample.h
- |—— role_side_back.cpp
- |—— role_side_back.h
- |—— role_side_forward.cpp
- |—— role_side_forward.h
- |—— role_side_half.cpp
- |—— role_side_half.h
- |—— sample_coach.cpp
- |—— sample_coach.h
- |—— sample_communication.cpp
- |—— sample_communication.h
- |—— sample_field_evaluator.cpp
- |—— sample_field_evaluator.h
- |—— sample_player.cpp
- |—— sample_player.h
- |—— sample_trainer.cpp
- |—— sample_trainer.h
- |—— soccer_role.cpp
- |—— soccer_role.h
- |—— start-debug.sh
- |—— start-offline.sh
- |—— start.sh.in
- |—— strategy.cpp
- |—— strategy.h
- |—— team_logo.xpm
- |—— train.sh.in
- |—— view_tactical.cpp
- |—— view_tactical.h

②コンパイル後

コンパイル後の agent2d は、ファイルとフォルダを合わせると 332 個、容量は 86.8MB

です。コンパイルによってファイルが 100 個くらい増えていますが、ほとんどは「sample_player-actgen_cross.o」といように拡張子が「.o」の機械語のファイルです。その他に増えたのは、「sample_player」などの実行ファイル、Makefile などのビルドに必要なファイルです。

- AUTHORS
- |— COPYING
- |— ChangeLog
- |— INSTALL
- |— Makefile
- |— Makefile.am
- |— Makefile.in
- |— NEWS
- |— NEWS.en
- |— README
- |— aclocal.m4
- |— autom4te.cache
- |— output.0
- |— output.1
- |— requests
- |— traces.0
- |— traces.1
- |— bootstrap
- |— config
- |— compile
- |— depcomp
- |— install-sh
- |— missing
- |— config.h
- |— config.h.in
- |— config.h.in~
- |— config.log
- |— config.status
- |— configure
- |— configure.ac
- |— m4

```

|   └── ax_boost_base.m4
|── src
|   ├── Makefile
|   ├── Makefile.am
|   ├── Makefile.in
|   ├── bhv_basic_move.cpp
|   ├── bhv_basic_move.h
|   ├── bhv_basic_offensive_kick.cpp
|   ├── bhv_basic_offensive_kick.h
|   ├── bhv_basic_tackle.cpp
|   ├── bhv_basic_tackle.h
|   ├── bhv_custom_before_kick_off.cpp
|   ├── bhv_custom_before_kick_off.h
|   ├── bhv_go_to_static_ball.cpp
|   ├── bhv_go_to_static_ball.h
|   ├── bhv_goalie_basic_move.cpp
|   ├── bhv_goalie_basic_move.h
|   ├── bhv_goalie_chase_ball.cpp
|   ├── bhv_goalie_chase_ball.h
|   ├── bhv_goalie_free_kick.cpp
|   ├── bhv_goalie_free_kick.h
|   ├── bhv_penalty_kick.cpp
|   ├── bhv_penalty_kick.h
|   ├── bhv_prepare_set_play_kick.cpp
|   ├── bhv_prepare_set_play_kick.h
|   ├── bhv_set_play.cpp
|   ├── bhv_set_play.h
|   ├── bhv_set_play_free_kick.cpp
|   ├── bhv_set_play_free_kick.h
|   ├── bhv_set_play_goal_kick.cpp
|   ├── bhv_set_play_goal_kick.h
|   ├── bhv_set_play_indirect_free_kick.cpp
|   ├── bhv_set_play_indirect_free_kick.h
|   ├── bhv_set_play_kick_in.cpp
|   ├── bhv_set_play_kick_in.h
|   └── bhv_set_play_kick_off.cpp

```

- | |—— bhv_set_play_kick_off.h
- | |—— bhv_their_goal_kick_move.cpp
- | |—— bhv_their_goal_kick_move.h
- | |—— chain_action
 - | |—— actgen_action_chain_length_filter.h
 - | |—— actgen_cross.cpp
 - | |—— actgen_cross.h
 - | |—— actgen_direct_pass.cpp
 - | |—— actgen_direct_pass.h
 - | |—— actgen_self_pass.cpp
 - | |—— actgen_self_pass.h
 - | |—— actgen_shoot.cpp
 - | |—— actgen_shoot.h
 - | |—— actgen_short_dribble.cpp
 - | |—— actgen_short_dribble.h
 - | |—— actgen_simple_dribble.cpp
 - | |—— actgen_simple_dribble.h
 - | |—— actgen_strict_check_pass.cpp
 - | |—— actgen_strict_check_pass.h
 - | |—— action_chain_graph.cpp
 - | |—— action_chain_graph.h
 - | |—— action_chain_holder.cpp
 - | |—— action_chain_holder.h
 - | |—— action_generator.h
 - | |—— action_state_pair.h
 - | |—— bhv_chain_action.cpp
 - | |—— bhv_chain_action.h
 - | |—— bhv_normal_dribble.cpp
 - | |—— bhv_normal_dribble.h
 - | |—— bhv_pass_kick_find_receiver.cpp
 - | |—— bhv_pass_kick_find_receiver.h
 - | |—— bhv_strict_check_shoot.cpp
 - | |—— bhv_strict_check_shoot.h
 - | |—— body_force_shoot.cpp
 - | |—— body_force_shoot.h
 - | |—— clear_ball.cpp

			clear_ball.h
			clear_generator.cpp
			clear_generator.h
			cooperative_action.cpp
			cooperative_action.h
			cross_generator.cpp
			cross_generator.h
			dribble.cpp
			dribble.h
			field_analyzer.cpp
			field_analyzer.h
			field_evaluator.h
			hold_ball.cpp
			hold_ball.h
			neck_turn_to_receiver.cpp
			neck_turn_to_receiver.h
			pass.cpp
			pass.h
			pass_checker.h
			predict_ball_object.h
			predict_player_object.h
			predict_state.cpp
			predict_state.h
			self_pass_generator.cpp
			self_pass_generator.h
			shoot.cpp
			shoot.h
			shoot_generator.cpp
			shoot_generator.h
			short_dribble_generator.cpp
			short_dribble_generator.h
			simple_pass_checker.cpp
			simple_pass_checker.h
			strict_check_pass_generator.cpp
			strict_check_pass_generator.h
			tackle_generator.cpp

- | | └── tackle_generator.h
- | └── coach.conf
- | └── communication.h
- | └── formations-dt
 - | | └── before-kick-off.conf
 - | | └── defense-formation.conf
 - | | └── defense-formation_1.conf
 - | | └── goal-kick-opp.conf
 - | | └── goal-kick-our.conf
 - | | └── goalie-catch-opp.conf
 - | | └── goalie-catch-our.conf
 - | | └── goalie-formation.conf
 - | | └── indirect-freekick-opp-formation.conf
 - | | └── indirect-freekick-our-formation.conf
 - | | └── kickin-our-formation.conf
 - | | └── normal-formation.conf
 - | | └── offense-formation.conf
 - | | └── setplay-opp-formation.conf
 - | | └── setplay-our-formation.conf
- | └── formations-keeper
 - | | └── before-kick-off.conf
 - | | └── defense-formation.conf
 - | | └── goal-kick-opp.conf
 - | | └── goal-kick-our.conf
 - | | └── goalie-catch-opp.conf
 - | | └── goalie-catch-our.conf
 - | | └── goalie-formation.conf
 - | | └── indirect-freekick-opp-formation.conf
 - | | └── indirect-freekick-our-formation.conf
 - | | └── kickin-our-formation.conf
 - | | └── normal-formation.conf
 - | | └── offense-formation.conf
 - | | └── setplay-opp-formation.conf
 - | | └── setplay-our-formation.conf
- | └── formations-taker
 - | | └── before-kick-off.conf

- | | |—— defense-formation.conf
- | | |—— goal-kick-opp.conf
- | | |—— goal-kick-our.conf
- | | |—— goalie-catch-opp.conf
- | | |—— goalie-catch-our.conf
- | | |—— goalie-formation.conf
- | | |—— indirect-freekick-opp-formation.conf
- | | |—— indirect-freekick-our-formation.conf
- | | |—— kickin-our-formation.conf
- | | |—— normal-formation.conf
- | | |—— offense-formation.conf
- | | |—— setplay-opp-formation.conf
- | | |—— setplay-our-formation.conf
- | |—— intention_receive.cpp
- | |—— intention_receive.h
- | |—— intention_wait_after_set_play_kick.cpp
- | |—— intention_wait_after_set_play_kick.h
- | |—— keepaway.sh
- | |—— keepaway.sh.in
- | |—— keepaway_communication.cpp
- | |—— keepaway_communication.h
- | |—— main_coach.cpp
- | |—— main_player.cpp
- | |—— main_trainer.cpp
- | |—— neck_default_intercept_neck.cpp
- | |—— neck_default_intercept_neck.h
- | |—— neck_goalie_turn_neck.cpp
- | |—— neck_goalie_turn_neck.h
- | |—— neck_offensive_intercept_neck.cpp
- | |—— neck_offensive_intercept_neck.h
- | |—— player.conf
- | |—— role_center_back.cpp
- | |—— role_center_back.h
- | |—— role_center_forward.cpp
- | |—— role_center_forward.h
- | |—— role_defensive_half.cpp

- | |—— role_defensive_half.h
- | |—— role_goalie.cpp
- | |—— role_goalie.h
- | |—— role_keepaway_keeper.cpp
- | |—— role_keepaway_keeper.h
- | |—— role_keepaway_taker.cpp
- | |—— role_keepaway_taker.h
- | |—— role_offensive_half.cpp
- | |—— role_offensive_half.h
- | |—— role_sample.cpp
- | |—— role_sample.h
- | |—— role_side_back.cpp
- | |—— role_side_back.h
- | |—— role_side_forward.cpp
- | |—— role_side_forward.h
- | |—— role_side_half.cpp
- | |—— role_side_half.h
- | |—— sample_coach
- | |—— sample_coach-main_coach.o
- | |—— sample_coach-sample_coach.o
- | |—— sample_coach.cpp
- | |—— sample_coach.h
- | |—— sample_communication.cpp
- | |—— sample_communication.h
- | |—— sample_field_evaluator.cpp
- | |—— sample_field_evaluator.h
- | |—— sample_player
- | |—— sample_player-actgen_cross.o
- | |—— sample_player-actgen_direct_pass.o
- | |—— sample_player-actgen_self_pass.o
- | |—— sample_player-actgen_shoot.o
- | |—— sample_player-actgen_short_dribble.o
- | |—— sample_player-actgen_simple_dribble.o
- | |—— sample_player-actgen_strict_check_pass.o
- | |—— sample_player-action_chain_graph.o
- | |—— sample_player-action_chain_holder.o

- | |—— sample_player-bhv_basic_move.o
- | |—— sample_player-bhv_basic_offensive_kick.o
- | |—— sample_player-bhv_basic_tackle.o
- | |—— sample_player-bhv_chain_action.o
- | |—— sample_player-bhv_custom_before_kick_off.o
- | |—— sample_player-bhv_go_to_static_ball.o
- | |—— sample_player-bhv_goalie_basic_move.o
- | |—— sample_player-bhv_goalie_chase_ball.o
- | |—— sample_player-bhv_goalie_free_kick.o
- | |—— sample_player-bhv_normal_dribble.o
- | |—— sample_player-bhv_pass_kick_find_receiver.o
- | |—— sample_player-bhv_penalty_kick.o
- | |—— sample_player-bhv_prepare_set_play_kick.o
- | |—— sample_player-bhv_set_play.o
- | |—— sample_player-bhv_set_play_free_kick.o
- | |—— sample_player-bhv_set_play_goal_kick.o
- | |—— sample_player-bhv_set_play_indirect_free_kick.o
- | |—— sample_player-bhv_set_play_kick_in.o
- | |—— sample_player-bhv_set_play_kick_off.o
- | |—— sample_player-bhv_strict_check_shoot.o
- | |—— sample_player-bhv_their_goal_kick_move.o
- | |—— sample_player-body_force_shoot.o
- | |—— sample_player-clear_ball.o
- | |—— sample_player-clear_generator.o
- | |—— sample_player-cooperative_action.o
- | |—— sample_player-cross_generator.o
- | |—— sample_player-dribble.o
- | |—— sample_player-field_analyzer.o
- | |—— sample_player-hold_ball.o
- | |—— sample_player-intention_receive.o
- | |—— sample_player-intention_wait_after_set_play_kick.o
- | |—— sample_player-keepaway_communication.o
- | |—— sample_player-main_player.o
- | |—— sample_player-neck_default_intercept_neck.o
- | |—— sample_player-neck_goalie_turn_neck.o
- | |—— sample_player-neck_offensive_intercept_neck.o

- | |—— sample_player-neck_turn_to_receiver.o
- | |—— sample_player-pass.o
- | |—— sample_player-predict_state.o
- | |—— sample_player-role_center_back.o
- | |—— sample_player-role_center_forward.o
- | |—— sample_player-role_defensive_half.o
- | |—— sample_player-role_goalie.o
- | |—— sample_player-role_keepaway_keeper.o
- | |—— sample_player-role_keepaway_taker.o
- | |—— sample_player-role_offensive_half.o
- | |—— sample_player-role_sample.o
- | |—— sample_player-role_side_back.o
- | |—— sample_player-role_side_forward.o
- | |—— sample_player-role_side_half.o
- | |—— sample_player-sample_communication.o
- | |—— sample_player-sample_field_evaluator.o
- | |—— sample_player-sample_player.o
- | |—— sample_player-self_pass_generator.o
- | |—— sample_player-shoot.o
- | |—— sample_player-shoot_generator.o
- | |—— sample_player-short_dribble_generator.o
- | |—— sample_player-simple_pass_checker.o
- | |—— sample_player-soccer_role.o
- | |—— sample_player-strategy.o
- | |—— sample_player-strict_check_pass_generator.o
- | |—— sample_player-tackle_generator.o
- | |—— sample_player-view_tactical.o
- | |—— sample_player.cpp
- | |—— sample_player.h
- | |—— sample_trainer
- | |—— sample_trainer-main_trainer.o
- | |—— sample_trainer-sample_trainer.o
- | |—— sample_trainer.cpp
- | |—— sample_trainer.h
- | |—— soccer_role.cpp
- | |—— soccer_role.h

```
|   |—— start-debug.sh
|   |—— start-offline.sh
|   |—— start.sh
|   |—— start.sh.in
|   |—— strategy.cpp
|   |—— strategy.h
|   |—— team_logo.xpm
|   |—— train.sh
|   |—— train.sh.in
|   |—— view_tactical.cpp
|   |—— view_tactical.h
|—— stamp-h1
|—— start.sh
```

(4) コンパイル

①コンパイル (ビルドの過程)

C 言語のソースファイルから実行ファイルを作ることをコンパイルと言います。コンパイルは、「プリプロセス」、「コンパイル」、「アセンブル」、「リンク」の4つの過程に分けることができます。詳しくは、下記のページをご覧ください。ちなみに、紹介したページの中にある「gcc」を「g++」にすれば、C++をコンパイルすることができます。

C 言語がコンパイルされて実行可能になるまでの流れ

<http://aoking.hatenablog.jp/entry/20121109/1352457273>

「ビルド」という作業は何を指しているのか

<https://www.atmarkit.co.jp/ait/articles/1105/23/news128.html>

ダイナミックリンクとスタティックリンク

<https://www.atmarkit.co.jp/ait/articles/1105/27/news111.html>

もっと詳しく知りたい方は、「C 言語本格入門～基礎知識からコンピュータの本質まで」の「1-4 C 言語の開発者ツールの役割」をご覧ください。

<http://gihyo.jp/book/2018/978-4-7741-9616-9>

②Make

ファイル数が少ない時はコマンドラインを使ってコンパイルをしても問題ありませんが、ファイル数が増えてくる何度もコマンドを打たなければならず大変です。そこで、**Makefile**を作成して **make** を使うと一度でコンパイルを実行することができます。

make コマンドを使ってみよう

<https://www.miraclelinux.com/tech-blog/0icygs>

make を使ってソフトウェアをビルドしてみよう

<https://www.atmarkit.co.jp/ait/articles/1106/07/news131.html>

Makefile をいろいろ書き換えながらビルドしてみよう

<https://www.atmarkit.co.jp/ait/articles/1106/10/news115.html>

③GNU Autotool

ファイル数が増えた場合や、自分以外の環境でビルドする必要がある時には「GNU Autotool」を使って **Makefile** を作成すると便利で、Agent2d も「GNU Autotool」を使って、ソースファイルをビルドしています。

「GNU Autotool」の使い方を簡単に説明すると以下のようになります。

1. 必要ファイルの追加

automake コマンド実行時に必要とされる、「INSTALL」、「NEWS」、「README」、「LICENSE」、「AUTHORS」、「ChangeLog」というファイルを作成します。設定によっては、これらのファイルを作らなくても実行することができます。

2. 「Makefile.am」 ファイルの作成

「**Makefile.am**」には、どのファイルを、どのような順番でコンパイルするかを記載します。こちらはコンパイルするファイルが入っているフォルダに分けて作成する必要があり、agent2dにおいては、トップフォルダと src フォルダの中に記載内容が異なる「**Makefile.am**」があります。src フォルダの中の「**Makefile.am**」には下記のような記載があり、どのファイルをコンパイルして実行ファイルが作られているかが分かります。

```
sample_player_SOURCES = ¥
                        $(CHAINACTION_SOURCES) ¥
                        $(PLAYERSOURCES)
sample_player_CXXFLAGS = -W -Wall
sample_player_LDFLAGS =
sample_player_LDADD =
```

Makefile.am の編集

<http://capm-network.com/?tag=Makefile.am%E3%81%AE%E7%B7%A8%E9%9B%86>

3. 「configure.ac」 ファイルの作成

「**configure.ac**」には、コンパイルを実行するときに必要な情報が記載されます。プロジェクトルートで「autoscan」を実行すると、「configure.scan」というファイルが作られます。このファイルに必要な修正を加えて、「configure.ac」を作成します。

configure.ac の編集

<http://capm-network.com/?tag=configure.ac%E3%81%AE%E7%B7%A8%E9%9B%86>

4. configure スクリプトの作成

「autoconf」を実行すると、configure スクリプトを作成できます。agent2d では、コンパイルに必要なファイルを追加したら、configure スクリプトを更新する必要があります。

「bootstrap」ファイルに configure スクリプトを更新するのに必要なコマンドが記載されていますので、「./bootstrap」を実行すると、configure スクリプトが更新できます。

ドキュメント/ファイルの追加・削除・リネーム

<http://rctools.osdn.jp/pukiwiki/index.php?%A5%C9%A5%AD%A5%E5%A5%E1%A5%F3%A5%C8/%A5%D5%A5%A1%A5%A4%A5%EB%A4%CE%C4%C9%B2%C3%A1%A6%BA%EF%BD%FC%A1%A6%A5%EA%A5%CD%A1%BC%A5%E0>

5. Makefile の作成

「./configure」を実行すると、Makefile を作成することができます。

「GNU Autotool」を実行すると、たくさんのファイルが作成されて面食らってしまいますが、重要なファイルは「Makefile.am」と「configure.ac」の2つだけですので、その2つのファイルを理解できれば問題ありません。

詳しい内容は、下記のサイトをご覧ください。

Autotools

<https://ja.wikipedia.org/wiki/Autotools>

autotools を使ってみよう

<https://www.miraclelinux.com/tech-blog/reqys8>

GNU Autotools で「Hello, World」

<https://qiita.com/narupo/items/f63b8e768f17ce50f398>

Autotools (automake, autoconf, libtool) 使い方まとめ

<http://tamaobject.hatenablog.com/entry/2013/08/01/165119>

④agent2d でファイルを追加した時

ドキュメント/ファイルの追加・削除・リネーム

<http://rctools.osdn.jp/pukiwiki/index.php?%A5%C9%A5%AD%A5%E5%A5%E1%A5%F3%A5%C8/%A5%D5%A5%A1%A5%A4%A5%EB%A4%CE%C4%C9%B2%C3%A1%A6%BA%EF%BD%FC%A1%A6%A5%EA%A5%CD%A1%BC%A5%E0>

⑤agent2d のライブラリをプログラムに同梱する方法

agent2d/公開用バイナリ作成方法

<http://rctools.osdn.jp/pukiwiki/index.php?agent2d/%B8%F8%B3%AB%CD%D1%A5%D0%A5%A4%A5%CA%A5%EA%BA%EE%C0%AE%CA%FD%CB%A1>

公開用パッケージの作り方

<http://rctools.osdn.jp/pukiwiki/index.php?cmd=read&page=agent2d%2F%B8%F8%B3%AB%CD%D1%A5%D1%A5%C3%A5%B1%A1%BC%A5%B8%A4%CE%BA%EE%A4%EA%CA%FD>

(5) src フォルダの中の「start.sh」の内容

各自が作成したメインフォルダの中にある「start.sh」→ src フォルダの中にある「start.sh」という流れで動いていきます。src フォルダの中にある「start.sh」はシェルスクリプトで書かれており、主な流れは下記の通りです。

①LIBPATH の書き込み (12 行~20 行)

共有ライブラリへのパスである LD_LIBRARY_PATH に LIBPATH に記載されたパスを追加しています。例えば、私の agent2d であれば、「LIBPATH=/home/mm/local/lib」を環境変数に追加します。このプログラムで疑問だったのは、なぜ私の PC のパスである「/home/mm/local/lib」を追加することによって、教室の PC で agent2d が動くのか、ということでした。

その理由としては、すでに教室の PC には「librcsc」の正しい環境パスが設定されていて、私の PC のパスが設定されるか否かは、プログラムを動かすのには関係ないということでした。あと、下記の部分が分かりにくかったので解説しておきます。

```
if [ x"$LIBPATH" != x ]; then
if [ x"$LD_LIBRARY_PATH" = x ]; then
LD_LIBRARY_PATH=$LIBPATH
```

\$LIBPATH と記載することによって、LIBPATH に入っているパスが展開されて表示されます。したがって、1 行目は、もし LIBPATH が空でなく、LD_LIBRARY_PATH が空っぽだった場合は、LD_LIBRARY_PATH に LIBPATH の内容を代入するという意味です。

②実行ファイル、conf ファイル、port 番号などを変数に入れる (24 行~53 行)

最初に「DIR=`dirname \$0`」の記載があります。「\$0」で実行したスクリプトのパス、「dirname」でそのスクリプトのパスの中から、ディレクトリ名を取得することができます。このプログラムを実行すると「\$0」が「./start.sh」であるため、「dirname」を実行すると「./」が DIR に代入されます。

その後は、下記のような変数にファイルへのパスや数字などをいれていきます。

```
player="${DIR}/sample_player"
coach="${DIR}/sample_coach"
teamname="HELIOS_base"
host="localhost"
port=6000
```

③ヘルプの内容表示（55 行～80 行）

ヘルプの内容は下記の通りです。

```
usage()
{
    (echo "Usage: $0 [options]"
    echo "Available options:"
    echo "      --help                prints this"
    echo "  -h, --host HOST            specifies server host (default: localhost)"
    echo "  -p, --port PORT            specifies server port (default: 6000)"
    echo "  -P  --coach-port PORT      specifies server port for online coach (default:
6002)"
    echo "  -t, --teamname TEAMNAME    specifies team name"
    echo "  -n, --number NUMBER        specifies the number of players"
    echo "  -u, --unum UNUM            specifies the uniform number of players"
    echo "  -C, --without-coach        specifies not to run the coach"
    echo "  -f, --formation DIR        specifies the formation directory"
    echo "  --team-graphic FILE        specifies the team graphic xpm file"
    echo "  --offline-logging           writes offline client log (default: off)"
    echo "  --offline-client-mode       starts as an offline client (default: off)"
    echo "  --debug                     writes debug log (default: off)"
    echo "  --debug-server-connect      connects to the debug server (default: off)"
    echo "  --debug-server-host HOST    specifies debug server host (default:
localhost)"
    echo "  --debug-server-port PORT    specifies debug server port (default: 6032)"
    echo "  --debug-server-logging      writes debug server log (default: off)"
    echo "  --log-dir DIRECTORY         specifies debug log directory (default: /tmp)"
    echo "  --debug-log-ext EXTENSION   specifies debug log file extension
(default: .log)"
    echo "  --fullstate FULLSTATE_TYPE specifies fullstate model handling"
    echo "                                FULLSTATE_TYPE is one of
[ignore | reference | override].") 1>&2
}
```


陸君がヘルプ部分を日本語化してくれたので、その部分を転載します。

```
(echo "使い方: $0 [オプション]"
echo "Available options:"
echo "      --help                ヘルプを表示します。"
echo "  -h, --host ホスト          サーバーに接続するホストを設定します。
例:192.168.1.0 デフォルト:localhost"

echo "  -p, --port ポート          サーバーに接続するポートを設定します。
例:1000 デフォルト:6000"

echo "  -P, --coach-port ポート      サーバーに接続するコーチポートを設定し
ます。例:1002 デフォルト:6002"

echo "  -t, --teamname チームネーム    チームネームを設定します。例：
HELIOS-BASE"

echo "  -n, --number 数字            サーバーに参加させる人数を設定します。
"
echo "  -u, --unum UNUM              specifies the uniform number of
players"

echo "  -C, --without-coach          specifies not to run the coach"
echo "  -f, --formation ファイルパス   フォーメーションのフォルダの場所を指
定します。"

echo "  --team-graphic ファイル        ログの指定をします。xpm 拡張子に対応
しています。"

echo "  --offline-logging            オフラインの時のログを書き込みます。デ
フォルト:off"

echo "  --offline-client-mode        ローカル内でサーバーを起動します デフ
ォルト:off"

echo "  --debug                    ログを書きます デフォルト:off"
```

echo " --debug-server-connect デフォルト:off"	デバッグサーバーに接続します デフォルト:off"
echo " --debug-server-host ホスト 定します。 デフォルト:localhost"	デバッグサーバーに接続するホストを設定します。 デフォルト:localhost"
echo " --debug-server-port ポート 定します。 デフォルト:6032"	デバッグサーバーに接続するポートを設定します。 デフォルト:6032"
echo " --debug-server-logging デフォルト:off"	デバッグサーバーのログを書き込みます。デフォルト:off"
echo " --log-dir ディレクトリ デフォルト:/tmp"	ログを書き込むフォルダを指定します。デフォルト:/tmp"
echo " --debug-log-ext 拡張子 ト:.log"	ログの拡張子を指定します。デフォルト:.log"
echo " --fullstate FULLSTATE_TYPE echo " [ignore reference override].") 1>&2 }	specifies fullstate model handline" FULLSTATE_TYPE is one of FULLSTATE_TYPE is one of

④引数の処理 (82 行~262 行)

最初にシェルスクリプトの規則を説明します。

1. 整数に関する評価演算子

- eq 等しい
- ne 等しくない
- lt より小さい
- le ~以下
- gt より大きい
- ge ~以上

文字列の比較、ファイル属性の評価演算子については、下記のページをご覧ください。

シェルスクリプト test コマンド 条件評価 スクリプトの書き方

http://bioinfo-dojo.net/2016/02/09/test_command_condition/

262 行まで下記のようなプログラムが続きます。

```

while [ $# -gt 0 ] # 引数が 0 よりも大きく
do
    case $1 in
        # 第 1 引数が

        --help)      # --help だったら help を表示する
            usage
            exit 0    # 正常終了
            ;;

        -h|--host)    # -h または --host だったら
            if [ $# -lt 2 ]; then # 引数が 2 よりも小さかったら
                usage            # ヘルプの内容を表示 (第 2 引数があるはずなので)
                exit 1          # 異常終了
            fi
            host="${2}" # 第 2 引数を host に代入する
            shift 1     # shift 1 で次の引数にシフトする
            ;;
    esac
done

```

⑤その他の処理 (264 行～)

⑥start.sh を実行した後にコンソールに表示される内容

最後の部分に表示された内容を見ると、割り当てられた 18 名の選手の中から 11 名の選手を選択し、適切なポジションに割り当てていることが分かります。

```

*****
HELIOS base
Created by Hidehisa Akiyama and Hiroki Shimora
Copyright 2000-2007.  Hidehisa Akiyama
Copyright 2007-2012.  Hidehisa Akiyama and Hiroki Shimora
All rights reserved.
*****

PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.037 ms

--- localhost ping statistics ---

```

1 packets transmitted, 1 received, 0% packet loss, time 0ms

rtt min/avg/max/mdev = 0.037/0.037/0.037/0.000 ms

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.

Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora

All rights reserved.

This program is based on agent2d created by Hidehisa Akiyama.

Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.

All rights reserved.

mochizuki2d: init ok. unum: 1 side: 1

mochizuki2d 1: KickTable created.

mochizuki2d 1: [-1, 0] set synch see mode.

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.

Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora

All rights reserved.

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.

Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora

All rights reserved.

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.

Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora

All rights reserved.

This program is based on agent2d created by Hidehisa Akiyama.

Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.

All rights reserved.

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.

Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora

All rights reserved.

mochizuki2d: init ok. unum: 2 side: 1

This program is based on agent2d created by Hidehisa Akiyama.

Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.

All rights reserved.

mochizuki2d: init ok. unum: 3 side: 1

This program is based on agent2d created by Hidehisa Akiyama.

Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.

All rights reserved.

mochizuki2d: init ok. unum: 4 side: 1

This program is based on agent2d created by Hidehisa Akiyama.

Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.

All rights reserved.

mochizuki2d: init ok. unum: 5 side: 1

mochizuki2d 3: KickTable created.

mochizuki2d 3: [-1, 0] set synch see mode.

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.

Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora

All rights reserved.

mochizuki2d 2: KickTable created.

mochizuki2d 2: [-1, 0] set synch see mode.

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.

Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora

All rights reserved.

mochizuki2d 5: KickTable created.

mochizuki2d 5: [0, 1] set synch see mode.

This program is based on agent2d created by Hidehisa Akiyama.

Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.

All rights reserved.

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.

Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora

All rights reserved.

mochizuki2d: init ok. unum: 6 side: 1

This program is based on agent2d created by Hidehisa Akiyama.

Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.

All rights reserved.

mochizuki2d: init ok. unum: 7 side: 1

This program is based on agent2d created by Hidehisa Akiyama.

Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.

All rights reserved.

mochizuki2d: init ok. unum: 8 side: 1

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.
Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora
All rights reserved.

mochizuki2d 4: KickTable created.
mochizuki2d 4: [-1, 0] set synch see mode.
mochizuki2d 4: [0, 2] missed last action?(1) last decision=[-1, 0]

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.
Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora
All rights reserved.

This program is based on agent2d created by Hidehisa Akiyama.
Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.
All rights reserved.

mochizuki2d 7: KickTable created.
mochizuki2d 7: [-1, 0] set synch see mode.
mochizuki2d: init ok. unum: 9 side: 1

librcsc 4.1.0

Copyright 2000 - 2007. Hidehisa Akiyama.
Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora
All rights reserved.

mochizuki2d 8: KickTable created.
mochizuki2d 6: KickTable created.
mochizuki2d 8: [0, 1]mochizuki2d 6: [-1, 0] set synch see mode.
set synch see mode.

This program is based on agent2d created by Hidehisa Akiyama.
Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.
All rights reserved.

```

mochizuki2d: init ok.  unum: 10 side: 1
exit good bye
mm@mm-
VirtualBox:~/t1$ *****

This program is based on agent2d created by Hidehisa Akiyama.
Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.
All rights reserved.
*****
*****

librcsc 4.1.0
Copyright 2000 - 2007. Hidehisa Akiyama.
Copyright 2007 - 2011. Hidehisa Akiyama and Hiroki Shimora
All rights reserved.
*****
*****

This program is based on agent2d created by Hidehisa Akiyama.
Copyright 2006 - 2011. Hidehisa Akiyama and Hiroki Shimora.
All rights reserved.
*****

mochizuki2d: init ok.  unum: 11 side: 1
mochizuki2d 9:  KickTable created.
mochizuki2d 9: [-1, 0] set synch see mode.
mochizuki2d coach: [-1, 0] recv (ok eye on)
mochizuki2d 10:  KickTable created.
mochizuki2d 10: [-1, 0] set synch see mode.
mochizuki2d 11:  KickTable created.
mochizuki2d 11: [-1, 0] set synch see mode.
id speed step inc  power  stam  karea
0 1.000   6  45.0 100.0  55.0  1.085
1 0.790   5  47.4 100.0  52.6  1.044
2 0.837   6  51.3 100.0  48.7  1.172
3 0.833   4  43.4 100.0  56.6  1.074
4 0.929   5  45.8 100.0  54.2  1.088
5 0.924   6  44.6 100.0  55.4  1.135
6 0.947   5  43.0 100.0  57.0  1.153
7 0.907   4  41.4 100.0  58.6  0.994

```


8	0.787	5	45.3	100.0	54.7	1.105
9	0.801	6	49.1	100.0	50.9	1.126
10	0.836	6	48.5	100.0	51.5	1.035
11	0.813	4	42.4	100.0	57.6	1.076
12	0.778	5	50.5	100.0	49.5	1.024
13	0.833	5	41.9	100.0	58.1	0.986
14	0.959	6	44.9	100.0	55.1	1.084
15	0.972	7	45.6	100.0	54.4	1.022
16	0.882	5	43.0	100.0	57.0	1.048
17	0.945	6	47.3	100.0	52.7	1.034

mochizuki2d coach: change player 1 to type 0

mochizuki2d coach: change player 11 to type 15

mochizuki2d coach: change player 2 to type 14

mochizuki2d coach: change player 3 to type 6

mochizuki2d coach: change player 10 to type 17

mochizuki2d coach: change player 9 to type 4

mochizuki2d coach: change player 6 to type 5

mochizuki2d coach: change player 4 to type 7

mochizuki2d coach: change player 5 to type 16

mochizuki2d coach: change player 7 to type 3

mochizuki2d coach: change player 8 to type 13

（６）代表的なオプション

①リモート接続（-h, --host HOST）

教室の PC で「ifconfig」と入力すると表示される IP アドレスの中から「192.168.1.XX」のアドレスを探す。

agnet2d の引数に、「-h 192.168.1.XX」を追加する。

②オフラインログ（--offline-logging ）

rcssserver から送られてくる、すべてのセンサメッセージを「/temp」に保存します。保存ファイルはプレイヤーごとに作られます。

3. フォーメーションの変更

(1) フォルダ構造

src フォルダの中に「formations-dt」というフォルダがあり、下記のような構造となっています (図1)。

(図1) formations-dt フォルダ構造

```
|   |—— formations-dt
|   |   |—— before-kick-off.conf
|   |   |—— defense-formation.conf
|   |   |—— goal-kick-opp.conf
|   |   |—— goal-kick-our.conf
|   |   |—— goalie-catch-opp.conf
|   |   |—— goalie-catch-our.conf
|   |   |—— goalie-formation.conf
|   |   |—— indirect-freekick-opp-formation.conf
|   |   |—— indirect-freekick-our-formation.conf
|   |   |—— kickin-our-formation.conf
|   |   |—— normal-formation.conf
|   |   |—— offense-formation.conf
|   |   |—— setplay-opp-formation.conf
|   |   |—— setplay-our-formation.conf
```

フォーメーションファイルには、テキストファイルを直接修正するものと、fedit2 (フォーメーション編集ツール) を使って修正するものがあります。

・直接修正するファイル

ファイルの最初に「Formation Static」と記載されている以下のファイルです。before-kick-off.conf、goal-kick-opp.conf、goal-kick-our.conf、goalie-catch-opp.conf、goalie-catch-our.conf です。before-kick-off.conf ファイルを修正すると、キックオフ時のポジションを修正できます。

・fedit2 を使って修正するファイル

ファイルの最初に「Formation DelaunayTriangulation 2」と記載されている、上記以外のファイルです。

fedit2 ダウンロードページ

<http://rctools.osdn.jp/pukiwiki/index.php?fedit2>

(2) fedit2 の使い方

端末で「fedit2」と打つと起動できます。使い方については、「RoboCup サッカー2D シミュレーション講習会@秋キャンプ 2011」のスライド 36 以降に記載されています。スライドは下記のページの「講習会」と記載してある部分の「occersim2d-slide.pdf」というファイルをダウンロードして下さい。

<http://rc-oz.osdn.jp/pukiwiki/index.php?cmd=read&page=Event%2F2011%2FCamp&word=2011>

ちなみに、fedit2 を使って「defense-formation.conf」というファイルを開くと 115 ヶ所のボールの位置に対する選手の場所を設定できます。ボールの位置に対する選手の場所を修正した場合は、メニューバーの中にある「Replace」ボタンを押すと、記録されます。「Replace」ボタンを押すことを忘れてしまうと記録されませんので、ボールの位置が変わるたびに「Replace」ボタンを押して下さい。

検討事項 (2019/4/7)

ロボカップサッカーシミュレーション 2D リーグ必勝ガイド

<https://jaist.dl.osdn.jp/rctools/46021/RoboCup2DGuideBook-1.0.pdf>

本書の 144 ページ以降に「FormationEditor」の使い方が記載されています。

FormationEditor を実行するには、以下のように `--editor-mode` オプションを付けて `soccerwindow2` を起動します。

```
$ soccerwindow2 --editor-mode
```

起動後、メニューから“New Formation”を選択すると、画面が図 3.4 のような状態になります。図 3.4 FormationEditor の実行画面ダイアログに表示されている役割の名前は、役割クラスで定義している名前に対応しています。必要に応じて変更してください。また、役割配分が望みのものと異なるのであれば、ダイアログを操作して変更してください。後は以下の手順を実行するだけです。

第 3 章 チーム開発

1. ボールを移動
2. プレイヤを移動
3. “Record” ボタンで訓練データ保存
4. “Train” ボタンで学習を実施
5. 訓練データ作成と学習を繰り返し実行

6. メニューから保存して終了

作成したフォーメーションのファイルは、サンプルチームのフォーメーションとして使用可能です。サンプルチームの Strategy クラスをそのまま使うなら、味方チームのキックイン、ボールが敵陣に存在する場合、ボールが自陣に存在する場合、の3種類のフォーメーションを作成することになります

この説明を読む限りでは、「train」というプロセスが必要になりそうです。こちらの説明は FormationEditor 1 なので、私たちが使用している FormationEditor 2 とは違うかもしれませんが、FormationEditor 2 にも「train」というボタンはありますが、このボタンを押すことによってどのような影響があるかはわかりません。このボタンを押すとどのようなことが起こるかが分かった方は教えて下さい。

(2019/8/21 追記)

「RoboCup サッカー2D シミュレーション講習会@秋キャンプ 2011」を読んだところ、「train」は使用しないとのことでした。

4. Librcsc の解説

詳細は、「ロボカップサッカーシミュレーション 2D リーグ必勝ガイド」の 46 ページに記載されている「Doxygen によるリファレンス生成」をご覧ください

(1) SoccerAgent クラス

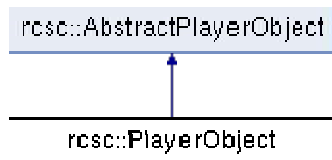


`CoachAgent`, `PlayerAgent`, `TrainerAgent` の 3 クラスの基本クラスとなっています。

(2) PlayerAgent クラス

`SoccerAgent` クラスを継承して作成。メンバ変数として `WorldModel` クラス、プレイヤーエージェント自身の情報を管理する `SelfObject` クラス、`PlayerObject` クラス、`BallObject` クラス、ボール所有者情報を判定する `InterceptTable` などをメンバ変数として持ち、`agent2d` を動かすときの基本となるクラスです。

(3) PlayerObject クラス

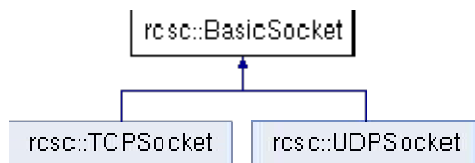


`AbstractPlayerObject` クラスを継承して作成し、他のプレイヤーの情報を管理するクラスです。

(4) WorldModel クラス

フィールド情報へアクセスする場合の窓口となるクラスです。

(5) BasicSocket クラス



`RcssServer` との通信を行うクラスです。

(6) AbstractPlayerObject クラス

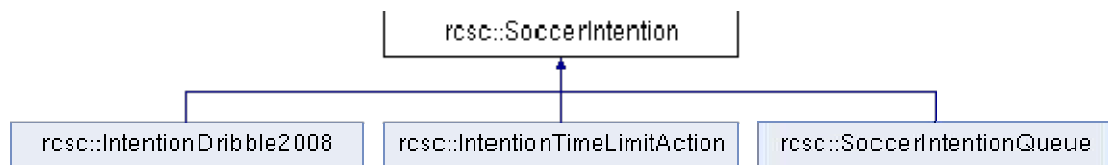


派生クラスに **PlayerObject** (他のプレイヤー)、**SelfObject** (自分) を指すときに使うクラスを持ちます。

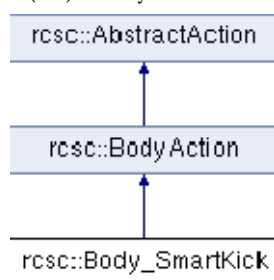
(7) InterceptTable クラス

ボールまで何サイクルでたどり付けるかを計算する関数などを提供するクラス。プレイやエージェントの意思決定においては、このボール補足サイクルを参照することによって、いずれのチームの誰がボールを持っているかを判断している。

(8) SoccerIntention クラス



(9) BodySmartkick クラス

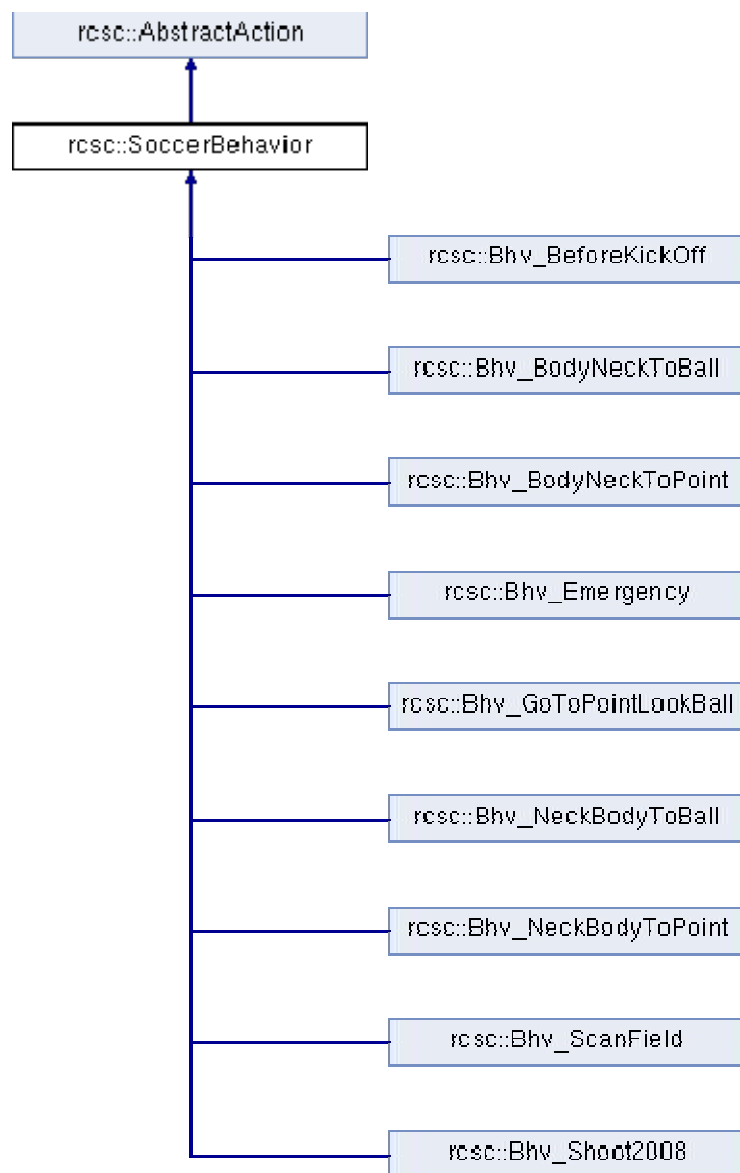


(10) CoachAgent クラス

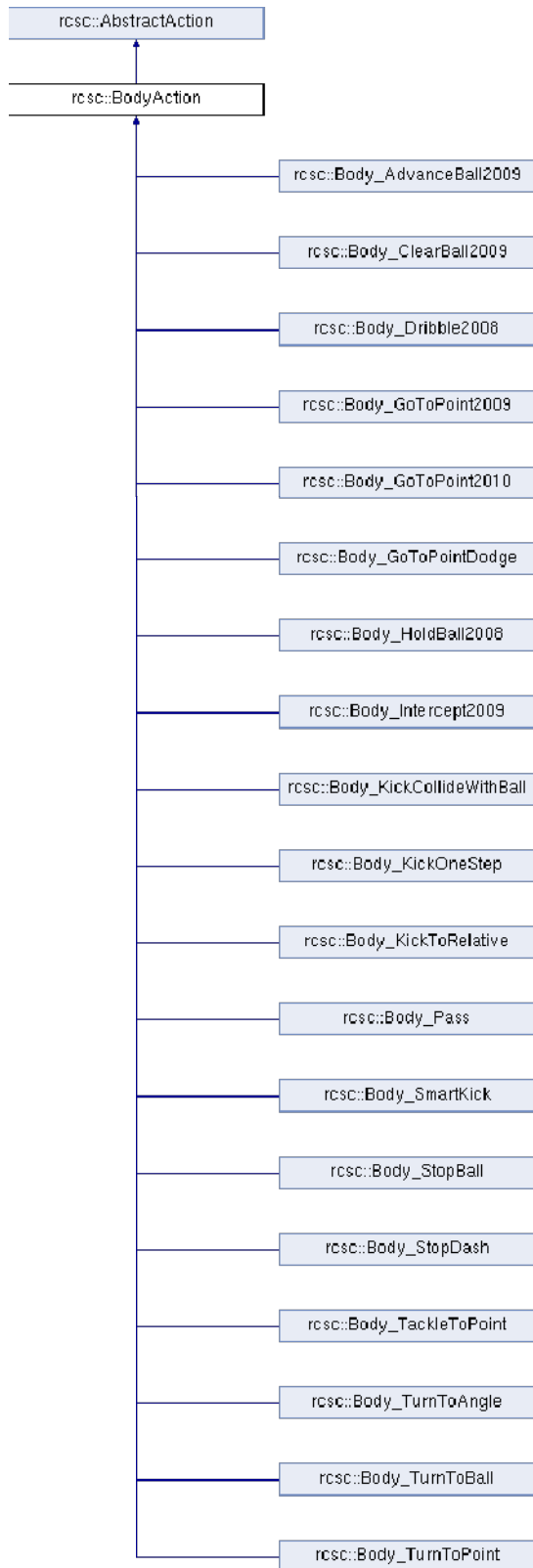
(11) AbstractAction クラス

継承図は Doxygen によるリファレンスを参照して下さい。

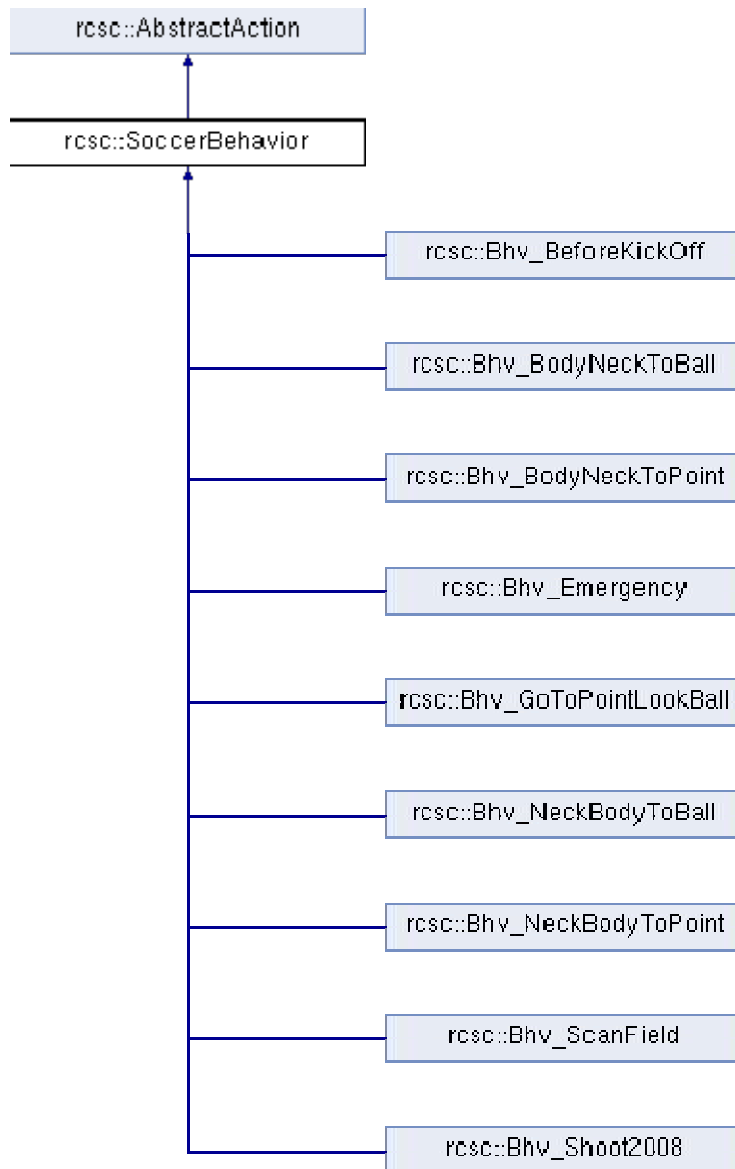
AbstractAction-SoccerBehavior クラス



(12) BodyAction クラス



(13) SoccerBehavior クラス



(14) BasicClient クラス

プログラムのメインループをラップしたクラスで、全てのサッカーエージェントの基本となるクラスとなっている (p 239)。

5. agent2d の解説

(1) プログラムの開始までの流れ

① トップフォルダの「start.sh」

トップフォルダの中に下記のような「start.sh」を作成しています。このファイルは `src` フォルダの「start.sh」を動かすためのプログラムであるため、`src` フォルダの「start.sh」の引数は、こちらのファイルに記載してください。

```
-----  
#!/bin/bash  
cd src  
./start.sh -t mochizuki2d #ここを変更  
# ./start.sh -t mochizuki2d --offline-logging  
# 引数で 「--offline-logging」と与えるとセンサ情報のログが残せる  
# 引数で 「-h 192.167.1.XX」と与えるとリモート接続できる  
-----
```

② src フォルダの「start.sh」

このファイルで、`src` フォルダの中にある「sample_player」という実行ファイルを動かします。

③ src フォルダの「main_player.cpp」

主要なインクルードファイル

```
#include <config.h>  
#include "sample_player.h"  
#include <rcsc/common/basic_client.h>
```

「sample_player」という実行ファイルは複数のファイルから作られていますが、最初に `main` 関数のある「main_player.cpp」から動きます。

最初に下記のヘッダファイルを読み込み、43 行目の「`SamplePlayer agent;`」で「agent」インスタンスを作成します。`SamplePlayer` クラスは `librcsc` の `Player agent` クラスの派生クラスであるため、`Player agent` クラスの関数などが使えます。プレイヤーの行動のパターンなどは、agent2d の `SamplePlayer` クラスではなく、`librcsc` の `Player agent` クラスの方で記述されており、`librcsc` の `player_agent.cpp` ファイルは 3,200 行ほどあります。

その後 58 行から始まるメイン関数で、sigaction（シグナルの動作の確認と変更）の設定をします 76 行の「rcsc::BasicClient client;」で client のインスタンスを作成し、問題がなければ 104 行の「client.run(&agent);」でクライアントを動かします。ちなみに、client インスタンスは、rcssserver との通信を行います。client インスタンスの詳細は、librcsc の basic_client.cpp に記載されています。

④世界モデル更新

サーバから届いた情報をどのように使って agent2d が意思決定を行っているかは、「ロボカップサッカーシミュレーション必勝ガイド」の 294～299 ページに記載されています。

⑤ログを見ながら選手の行動を分析する

こちらは、敵のゴール前で 10 番が 11 番にパスをする時のプレイヤー 10 番のログです。3647、3648 サイクルでパスの相手の 11 番を探し、3649 サイクルで 11 番にパスを出しています。ちなみに、ログの出し方は「RoboCup サッカー2D シミュレーションリーグ解説：サンプルエージェントを使ったチーム開発」をご覧ください。

https://www.jstage.jst.go.jp/article/jsoft/23/6/23_838/article/-char/ja/

「分析用ログ.txt」の 3647 ステップについて解析します。実際のログはこの PDF ファイルが保存されているファイルの中の「分析用ログ.txt」をご覧ください。以下に行数を記載しますが、行数は 1 行の表示文字数によって変わってきます。

1～27 行

サーバからの情報をもとに、ワールドモデルのアップデートを行っています。

28～58 行

味方と敵の選手のポジションを分析し、1 番近い選手と 2 番目に近い選手を推定します。

59～82 行

シチュエーションに応じてフォーメーションを決定しています。

83 行

Sample_player.cpp の doPreProcess 関数の実行

84 行～96 行

chain_action/bhv_normal_dribble.cpp を実行してドリブルを行う

97 行～110 行

ボールのスキャンや首振りを行う

111 行～113 行

コミュニケーションを行う

114 行

サーバにコメントを送る

115 行～130 行

ボールのポジションなどを再度アップデートして、サイクルを終了する。

(2) 全体のコントロール (src/sample_player.cpp)

src フォルダの中の「sample_player.cpp」に記載されています。このプログラムでは、src フォルダの中にある以下のヘッダファイルを読み込んでいます。

```
#include "sample_player.h"
#include "strategy.h"
#include "field_analyzer.h"
#include "action_chain_holder.h"
#include "sample_field_evaluator.h"
#include "soccer_role.h"
#include "sample_communication.h"
#include "keepaway_communication.h"
#include "bhv_penalty_kick.h"
#include "bhv_set_play.h"
#include "bhv_set_play_kick_in.h"
#include "bhv_set_play_indirect_free_kick.h"
#include "bhv_custom_before_kick_off.h"
#include "bhv_strict_check_shoot.h"
#include "view_tactical.h"
#include "intention_receive.h"
```

SamplePlayer クラスは、rcsc の PlayerAgent クラスの派生クラスになっています。

```
class SamplePlayer
{
public:
    : public rcsc::PlayerAgent {
```

クラスのメンバ変数として、下記の3つのクラスのポインタを持っています。

private:

```
Communication::Ptr M_communication;  
FieldEvaluator::ConstPtr M_field_evaluator;  
ActionGenerator::ConstPtr M_action_generator;
```

167 行～

```
bool SamplePlayer::initImpl( CmdLineParser & cmd_parser )
```

インスタンスが作成できたかのチェックを行っている

223 行～

```
void SamplePlayer::actionImpl()
```

SamplePlayer クラスはコンストラクタを作成すると、世界モデルのアップデートなどが終わった後に、244 行の 「 if (doPreprocess())」 が True となれば終了して、呼び出し元にもどる。それ以外の場合は、アクションチェーンのアップデートやカレントロールの作成などを行う。現在のプログラムではカレントロールの部分はコメントアウトされている。詳しくは () プレイヤーの役割を参照。

234 行～

```
// prepare action chain
```

```
//
```

```
M_field_evaluator = createFieldEvaluator();
```

```
M_action_generator = createActionGenerator();
```

```
①ActionChainHolder::instance().setFieldEvaluator( M_field_evaluator );
```

```
②ActionChainHolder::instance().setActionGenerator( M_action_generator );
```

①でフィールドの状況を得点化して、ActionChainHolder::instance()に入れる。

508 行～

```
bool SamplePlayer::doPreprocess()
```

586 行～ 行動のチェック

1. シュートが打てるかチェック、

2. 連続的な動作（ドリブルなど）ができるかチェック、
3. 強制的に蹴るかチェック
4. パスの受取のメッセージがないかチェック

625 行

`bool SamplePlayer::doShoot()`

条件がそろったら、`Bhv_StrictCheckShoot().execute(this)` を行う。

765 行～

```
FieldEvaluator::ConstPtr
```

```
SamplePlayer::createFieldEvaluator() const
```

```
{
```

```
    return FieldEvaluator::ConstPtr( new SampleFieldEvaluator );
```

```
}
```

「createFieldEvaluator()」を実行すると、「SampleFieldEvaluator」のインスタンスが作成されるため、その状態のポインタが返されると考えられる。

(3) ボールキックの行動評価関数 (sample_field_evaluator.cpp)

src フォルダの中の「sample_field_evaluator.cpp」に記載されており、以下のヘッダファイルを読み込んでいます。下記に記載しているヘッダファイルは、agent2d の中のファイルに限定しており、量が多くなってしまうため libresc のヘッダファイルは記載していません。

```
#include "sample_field_evaluator.h"
```

```
#include "field_analyzer.h"
```

```
#include "simple_pass_checker.h"
```

ちなみに、C++のプログラムでは、関数のプロトタイプ宣言や構造体の定義などを直接「.cpp」ファイルに記載するのではなく、ヘッダファイルに記載することによって、他のプログラムファイルから使用できるようにすることが多くなっています。そのため、「.h」と対になった「.cpp」ファイルがあり、「field_analyzer.h」、「field_analyzer.cpp」、「simple_pass_checker.h」、「simple_pass_checker.cpp」などの対になったファイルが src フォルダの中の「chain_action」フォルダの中にあります。

80 行～

```
double
```

```
SampleFieldEvaluator::operator()( const PredictState & state,
```

```
                                const std::vector< ActionStatePair > & /*path*/ )
```

```
const
```

```
{
```

```
    const double final_state_evaluation = evaluate_state( state );
```

```
//
```



```

// ???
//

double result = final_state_evaluation;

return result;
}

```

この関数を実行すると、100 行以降に記述されている「`evaluate_state(const PredictState & state)`」が呼び出され、その部分で計算した点数が返されます。

174 行～

ポイントの基本点数はボールの X 座標で、ゴールに近ければ近いほど基本のポイントがアップし、自分がシュートを打てる状態なら 100 万ポイントのボーナスが、ボールを持っていたら 50 万点がボーナスとして加算されます。

(4) ポジショニング動作 (bhv_basic_move.cpp)

src フォルダの中の「bhv_basic_move.cpp」に記載されています。以下のヘッダファイルを読み込んでいます。これらのファイルは src フォルダの中にあります。

```

#include "bhv_basic_move.h"
#include "strategy.h"
#include "bhv_basic_tackle.h"

```

66 行～ タックルの動作

78 行～

(5) プレイヤーの役割 (soccer_role.cpp)

src フォルダの中の「soccer_role.cpp」に記載されています。このファイルの中では、「soccer_role.h」以外の他のポジションのファイルは読み込んでいないため、作り込めばポジションごとに動きを変えることができます。ちなみに、「(6) (strategy.cpp)」では、他のポジションのファイルも読み込んでいるため、現在の状態ではポジション別の動きができるのか否かが不明なのですが、「(6) (strategy.cpp)」と「soccer_role.cpp」を読み比べて、そのあたりのことが分かったら教えて下さい。デフォルト実装は下記の通りです。

(追記)

role_center_forward.cpp ファイルの中で「std::cout」を使ってデバッグ情報を記載したところ、コンソールに文字が表示されたため、現在の状態でも役割別のファイルにプログラムを記載すれば別の動きをさせることができます。

```
bool
RoleDefensiveHalf::execute( PlayerAgent * agent )
{
    bool kickable = agent->world().self().isKickable();
    if ( agent->world().existKickableTeammate()
        && agent->world().teammatesFromBall().front()->distFromBall()
        < agent->world().ball().distFromSelf() )
    {
        kickable = false;
    }

    if ( kickable )
    {
        doKick( agent );
    }
    else
    {
        doMove( agent );
    }

    return true;
}
```

```

/*-----*/
/*!

*/
void
RoleDefensiveHalf::doKick( PlayerAgent * agent )
{
    if ( Bhv_ChainAction().execute( agent ) )
    {
        dlog.addText( Logger::TEAM,
                      __FILE__": (execute) do chain action" );
        agent->debugClient().addMessage( "ChainAction" );
        return;
    }

    Bhv_BasicOffensiveKick().execute( agent );
}

/*-----*/
/*!

*/
void
RoleDefensiveHalf::doMove( PlayerAgent * agent )
{
    Bhv_BasicMove().execute( agent );
}

```

(6) 戦略 (strategy.cpp)

src フォルダの中の「strategy.cpp」に記載されています。このファイルの中でフォーメーションのポジションと各選手の役割などを読み込んでいます。このプログラムでは、src フォルダの中にある以下のヘッダファイルを読み込みんでいます。初期設定では「USE_GENERIC_FACTORY」を定義していない限りすべてのポジションファイルを読み込んでいるため、各ポジションのファイルにプログラムを記載すればポジションごとに別の動きができると思います。ちなみに、USE_GENERIC_FACTORY」を定義すると多少動きが変わるようです。

```
#ifndef USE_GENERIC_FACTORY
#include "role_sample.h"
#include "role_center_back.h"
#include "role_center_forward.h"
#include "role_defensive_half.h"
#include "role_goalie.h"
#include "role_offensive_half.h"
#include "role_side_back.h"
#include "role_side_forward.h"
#include "role_side_half.h"
```

フォーメーションの切替や、ボールエリアなどが指定されています。600 行以降でカレントシチュエーションが、オフフェンスかディフェンスかノーマルかの判断を行い、794 行以降でシチュエーションに合わせてフォーメーションを切り替えています。

970 行以降 Strategy::get_ball_area(const Vector2D & ball_pos)関数で、ボールのポジションを基準として、「BA_Cross、BA_ShootChance、BA_DribbleAttack」などの、エリア分けをしているため、こちらのポジションによって行動を変更できると思われます。1130 行以降で、スタミナを考慮しながらダッシュのスピードを調整しています。

(7) コミュニケーションルール(sample_communication.cpp)

src フォルダの中の「sample_communication.cpp」に記載されています。このプログラムでは、src フォルダの中にある以下のヘッダファイルを読み込んでいます。読み込んでいるファイルが「strategy.h」なので、フォーメーションを作った後に、それにふさわしいコミュニケーションルールを作る必要があることが分かります。

```
#include "sample_communication.h"
```

```
#include "strategy.h"
```

このファイルを見ると、多く種類のメッセージを「addSayMessage」を使って送ることができます。「⑤ログを見ながら選手の行動を分析する」の部分で紹介した10番の3647ステップのログの最初の部分を見ると、8番から圧縮したメッセージを受け取り、それを解凍していることが分かります。メッセージの内容はボールの場所とゴールキーパーの位置のようです。ログを見るとスタミナの量などの様々な種類のメッセージを送っていることが分かりますが、どういう条件でそのメッセージを送っているのかは分かっていません。

```
3647 2 M ===receive hear [(hear 3647 -82 our 8 "G<0fzY25dH")]  
3647 2 M audio_sensor.cpp (parsePlayerMessage) clear old data
```

```
3647 2 M BallGoalieMessageParser: success! sender = 8 bpos(50.1 15.7) bvel(-  
0.2 -0.4) gpos(51.3 6.1) gbody -171.0
```

```
3647 4 M audio_memory.cpp: set heard ball: sender=8 pos=(50.100, 15.700)  
vel=(-0.24, -0.43)
```

```
3647 4 M audio_memory.cpp: set heard goalie: sender=8 pos=(51.30, 6.10) body=-  
171.0
```

(8) アクションチェイン

```
chain_action/bhv_strict_check_shoot.cpp: no shoot course
```

```
1361 128 M chain_action/bhv_normal_dribble.cpp: (finished). not finished yet.
```

```
1361 128 M chain_action/bhv_normal_dribble.cpp: (intention:execute) turn=0 dash=2
```

```
1361 256 M chain_action/strict_check_pass_generator.cpp (generate) PROFILE  
passer=self size=0/995 D=0 L=0 T=0 elapsed 1.413000 [ms]
```

```
1361 512 M chain_action/cross_generator.cpp (generate) first point(-8.5 25.9) is too far  
from the goal.
```

```
1361 128 M chain_action/short_dribble_generator.cpp: (generate) PROFILE size=28/27  
elapsed 0.024 [ms]
```

```
1361 128 M chain_action/self_pass_generator.cpp: (generate) PROFILE size=32/80  
elapsed 0.085 [ms]
```

```
1361 1048576 M >>>>> best chain:
```

①chain_action /action_generator.h

49～

ActionGenerator クラスの定義がされている

78～

こちらの generate 関数は純粋仮想関数となっている。

```
virtual
    void generate( std::vector< ActionStatePair > * result,
                  const PredictState & state,
                  const rsc::WorldModel & wm,
                  const std::vector< ActionStatePair > & path ) const = 0;
};
```

91 行～最後 ActionGenerator クラスから派生した、CompositeActionGenerator のクラスの定義がされている。

```
class CompositeActionGenerator
    : public ActionGenerator {
private:
    //! generator container
    std::vector< ConstPtr > M_generators;

public:

    void addGenerator( const ActionGenerator * g )
    {
        if ( g )
        {
            M_generators.push_back( ConstPtr( g ) );
        }
    }

    void generate( std::vector< ActionStatePair > * result,
                  const PredictState & state,
```

```

        const rsc::WorldModel & wm,
        const std::vector< ActionStatePair > & path ) const
    {
        for ( std::vector< ConstPtr >::const_iterator g = M_generators.begin();
              g != M_generators.end();
              ++g )
        {
            (*g)->generate( result, state, wm, path );
        }
    }
};

```

CompositeActionGenerator クラスは「sampleplayer.cpp」クラスの 789 行以下でインスタンスが使われている。ActGen_MaxActionChainLengthFilter 関数は、actgen_action_chain_length_filtter.h で定義されている・

```

ActionGenerator::ConstPtr
SamplePlayer::createActionGenerator() const
{
    CompositeActionGenerator * g = new CompositeActionGenerator();

    //
    // shoot
    //
    g->addGenerator( new ActGen_RangeActionChainLengthFilter
                    ( new ActGen_Shoot(),
                      2, ActGen_RangeActionChainLengthFilter::MAX ) );

    //
    // strict check pass
    //
    g->addGenerator( new ActGen_MaxActionChainLengthFilter
                    ( new ActGen_StrictCheckPass(), 1 ) );

    //
    // cross

```

```

//
g->addGenerator( new ActGen_MaxActionChainLengthFilter
                  ( new ActGen_Cross(), 1 ) );

//
// direct pass(コメントアウトを外した)
//
g->addGenerator( new ActGen_RangeActionChainLengthFilter
                  ( new ActGen_DirectPass(),
                    2, ActGen_RangeActionChainLengthFilter::MAX ) );

//
// short dribble
//
g->addGenerator( new ActGen_MaxActionChainLengthFilter
                  ( new ActGen_ShortDribble(), 1 ) );

//
// self pass (long dribble)
//
g->addGenerator( new ActGen_MaxActionChainLengthFilter
                  ( new ActGen_SelfPass(), 1 ) );

//
//simple dribble(コメントアウトを外した)
//
g->addGenerator( new ActGen_RangeActionChainLengthFilter
                  ( new ActGen_SimpleDribble(),
                    2, ActGen_RangeActionChainLengthFilter::MAX ) );

return ActionGenerator::ConstPtr( g );
}

```


②chain_action /action_chain_graph.cpp

116～ コンストラクタ

```
ActionChainGraph::ActionChainGraph( const FieldEvaluator::ConstPtr & evaluator,
                                     const ActionGenerator::ConstPtr & generator,
                                     unsigned long max_chain_length,
                                     long max_evaluate_limit )
: M_evaluator( evaluator ),
  M_action_generator( generator ),
  M_chain_count( 0 ),
  M_best_chain_count( 0 ),
  M_max_chain_length( max_chain_length ),
  M_max_evaluate_limit( max_evaluate_limit ),
  M_result(),
  M_best_evaluation( -std::numeric_limits< double >::max() )
{
#ifdef DEBUG_PAINT_EVALUATED_POINTS
    S_evaluated_points.clear();
    S_evaluated_points.reserve( DEFAULT_MAX_EVALUATE_LIMIT + 1 );
#endif
}
```

254～ doSearch 関数

③chain_action /bhv_chain_action.cpp

164～ execute 関数

シュート、ドリブルなどのコーポレートアクションの内容によって、必要な関数を実行する。

(9) シュート

①chain_action/shoot_generator.cpp

62 行～

シュートコースは最大 3 2 コース

334～

createShoot 関数でシュートコースを作成する

436～

maybeGoalieCatch 関数

ゴールキーパーがキャッチできるかのチェック

610～

opponetCanReach 関数

敵がブロックできるかのチェック

747～

evaluteCourses

シュートコースを評価して、最も得点の高いコースを決めている。

②chain_action/bhv_strict_check_shoot.cpp

shoot_generator.cpp で作成したベストコースを打つことに問題がないかのチェックを行っている。シュートに関する項目については、ほとんど shoot_generator.cpp で決めており、こちらではシュートを打てるかの最終チェックのみをしている。

(1 0) パス

①chain_action/self_pass_generator.cpp

85～

パスコースは最大 128 行

119～

generate 関数

145 行の createCourses 関数でパスコースを作った後で、147 行で sort を使ってゴールからの距離でソートしている。

164～

createCourses 関数

281 行の checkOpponent 関数でひっかかた場合は、パスコース自体が作られない

574 から最後まで

checkOpponent 関数

パスコースを「self_pass_generator.cpp」関数で作成しているが、「strict_check_pass_generator.cpp」でパスコースをもう一度作っているため、こちらのコースが使われているかは不明。

②chain_action/strict_check_pass_generator.cpp

183～

generate 関数

203 updatePasser(wm)

213 updateReceivers(wm)

222 updateOpponents(wm)

224 createCourses(wm)

226 行で sort を使ってゴールからの距離でソートしている。

484～

createCourses 関数

ダイレクトパス、リーディングパス、スループスのパスタイプ別にパスコースを作成する。

519～

createDirectPass 関数

657～

createLeadingPass 関数

875～

createThrouPass 関数

1149～

createPassCommon 関数

1543～

predictOpponentReachStep 関数

1590 行のリーチステップを 1 減らしている

1598～

predictOpponetReachStep 関数

1678、1801 行のリーチステップを 1 減らしている

③chain_action/bhv_pass_kick_find_receiver.cp

963～

doSayPass 関数

993 行でパス相手の背番号、レシーブポジション、次のボールのポジション、ボールのスピードを伝えている。

(1 1) ドリブル

参考情報

(1) Agent2d 関連

秋山さんホームページ

<http://rctools.osdn.jp/pukiwiki/>

秋山さん論文紹介ページ

<http://resweb2.jhk.adm.fukuoka-u.ac.jp/FukuokaUnivHtml/info/5687/R110J.html>

The RoboCup Soccer Simulator

<https://rcsoccersim.github.io/>

(2) 論文等

RoboCup サッカーにおけるニューラルネットワークによる評価関数モデリング

<http://www.osaka-kyoiku.ac.jp/~challeng/SIG-Challenge-051/SIG-Challenge-051-02.pdf>

RoboCup サッカー2D シミュレーションに対するゲームログからの評価関数の設定

https://www.jstage.jst.go.jp/article/fss/30/0/30_562/_pdf

RoboCup サッカーシミュレーションリーグ 2D における 局面評価関数の学習

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=2ahUKEwiVhpSE4JLkAhWOxYsBHfKeAuoQFjADegQICBAC&url=https%3A%2F%2Fipsj.ixsq.nii.ac.jp%2Ffej%2Findex.php%3Faction%3Dpages_view_main%26active_action%3Drepository_action_common_download%26item_id%3D95820%26item_no%3D1%26attribute_id%3D1%26file_no%3D1%26page_id%3D13%26block_id%3D8&usg=AOvVaw0IDSNtsMEeFbt6fXMc5wmW

RoboCup サッカーにおけるセットプレイのマーク割当手法の性能調査

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=10&ved=2ahUKEwiVhpSE4JLkAhWOxYsBHfKeAuoQFjAJegQIAhAC&url=https%3A%2F%2Fjsai.ixsq.nii.ac.jp%2Ffej%2F%3Faction%3Drepository_uri%26item_id%3D1380%26file_id%3D1%26file_no%3D1&usg=AOvVaw0TmsHxXsVzX0gUifn6M1dV

エージェント配置問題における三角形分割を利用した近似モデル

https://www.jstage.jst.go.jp/article/tjsai/23/4/23_4_255/_pdf

自己組織化マップを用いた教師あり学習 によるエージェント協調動作獲得手法

https://www.google.com/url?sa=t&ret=j&q=&esrc=s&source=web&cd=5&ved=2ahUKEwjK69u_45LkAhXUzIsBHfTOBBAQFjAEegQIABAC&url=https%3A%2F%2Fipsj.ixsq.nii.ac.jp%2Ffej%2F%3Faction%3Drepository_action_common_download%26item_id%3D70418%26item_no%3D1%26attribute_id%3D1%26file_no%3D1&usg=AOvVaw3ah_PAPhZw9Pdo7J5z-2eP

(3) 世界大会出場チームのアルゴリズム説明 (Team Description Paper)

https://wrighteagle2d.github.io/robocup_tdp.html

(4) C、C++関連

(概要)

C 言語

<https://ja.wikipedia.org/wiki/C%E8%A8%80%E8%AA%9E>

C++

<https://ja.wikipedia.org/wiki/C%2B%2B>

非情報系学生のための C/C++ 入門

<https://brain.cc.kogakuin.ac.jp/~kanamaru/lecture/prog1/index.html>

(参考書籍)

C 言語本格入門～基礎知識からコンピュータの本質まで

<http://gihyo.jp/book/2018/978-4-7741-9616-9>

スラスラわかる C++ 第2版

<https://www.shoeisha.co.jp/book/detail/9784798153872>

(5) Git、Github

①参考情報

Git/GitHub レベル別オススメ学習サイトまとめ完全保存版【2019.03】

<https://qiita.com/think-a-lot/items/b3c2e9060f46f5d4ea46>

今日からはじめる GitHub ～ 初心者が Git をインストールして、プルリクできるようになるまでを解説

<https://employment.en-japan.com/engineerhub/entry/2017/01/31/110000>

こっそり始める Git/GitHub 超入門

<https://www.itmedia.co.jp/author/207881/>

Git でやらかした時に使える 19 個の奥義

<https://qiita.com/muran001/items/dea2bbbbaea1260098051>

もう怖くない Git！チーム開発で必要な Git を完全マスター(udemy)

https://www.udemy.com/unscared_git/learn/v4/overview

ちなみに、「GitHub Desktop」というコマンド入力をしなくても良いアプリもありますので、興味のある方はそちらをご覧ください。

いよいよ登場！初心者こそ知っておきたい GitHub Desktop の使い方

<https://ferret-plus.com/8498>

②プログラムのバックアップ方法

プログラムのバージョン管理を行った方が良いでしょうので、バックアップに必要なコマンドを説明します。コマンドは、こちらのページの方法に従ってローカルリポジトリを作ったという前提で説明します。ちなみに私は、こちらの方法でローカルリポジトリを作った後で、agent2d のファイルを追加しました。

今日からはじめる GitHub ～ 初心者が Git をインストールして、プルリクできるようになるまでを解説

<https://employment.en-japan.com/engineerhub/entry/2017/01/31/110000>

1. ファイルをステージに追加

「git add」コマンドで修正したファイルをステージに追加することができます。ファイル名を指定してステージに追加することもできますが、「.」でカレントフォルダの中で変更したファイル全てを一度にステージに追加した方が簡単です。

```
git add .
```

2. ファイルをコミット

```
git commit
```

3. ローカルリポジトリを GitHub（リモートリポジトリ）と同期

```
git push
```

4. GitHub（リモートリポジトリとローカルリポジトリを同期

「git push」した時点でローカルリポジトリとリモートリポジトリは同一になっているはずですが、念のためリモートからローカルの方で同期をしておきます。

```
git pull
```

5. 以前のファイルへの戻し方

（1）ファイルの変更の取り消し(前回のコミットまで戻す)

```
git checkout -- training.txt
```

すべてのファイルの変更の取り消し

```
git checkout -- .
```

（2）git add（ステージした変更の取り消し（ファイルはそのまま）

①git add だけを取り消す

```
git rm --cached training.txt
```

②前回のコミットまで戻す

```
git reset HEAD training.txt
```

「git reset HEAD」を行った後で、「git checkout --」を行うと、

git add した元ファイルの変更を取り消し、
ファイルを前回のコミットの状態に戻すことができる。

「git reset」はリポジトリの内容で、ステージの内容を書き換える。
「git checkout -- .」はステージの内容にワークツリーの内容を書き換える。

「HEAD」は、今自分のいるブランチの最新のコミットを指している。

(3) 直前のコミットをやり直す

git commit --amend

このコマンドを入力すると、現在のステージの内容に直前のコミットを修正することができる。

ただし、このコマンドを入力すると、リモートリポジトリとローカルリポジトリの最終コミットが
異なってしまうため、リモートにプッシュしたコミットには使わない。

リモートにプッシュしてしまった場合は、直前のコミットをやり直すのではなく、
次のコミットを作って修正するようにする。

git log

```
-----  
commit 183d8213c229b79eacf7886c07e75e8f0fd55523  
Author: mmochizuki <mmocchi@pop07.odn.ne.jp>  
Date:   Fri Apr 19 09:58:55 2019 +0900
```

2 回めのコミット--メッセージの修正

```
commit 58df005aff077fff0e5ad092e0c2d88b700b516b  
Author: mmochizuki <mmocchi@pop07.odn.ne.jp>  
Date:   Fri Apr 19 09:54:44 2019 +0900
```

最初のコミット

(4) 以前のコミットまで戻す (戻したコミット以降のコミットは消滅する)

```
git reset --hard 58df005aff077fff0e5ad092e0c2d88b700b516b
```

HEAD is now at 58df005 最初のコミット

```
mm@mm-VirtualBox:~/work/Git_training$ git log
```

```
commit 58df005aff077fff0e5ad092e0c2d88b700b516b
```

```
Author: mmochizuki <mmocchi@pop07.odn.ne.jp>
```

```
Date:   Fri Apr 19 09:54:44 2019 +0900
```

git commit を取り消して元に戻す方法、徹底まとめ

<http://www-creators.com/archives/1116>

(5) リモートリポジトリのコミットを戻す

直接戻す方法はないようですので、下記の方法を使って戻してください。

Git でリモートリポジトリを巻き戻す

<https://qiita.com/rch1223/items/9377446c3d010d91399b>

[git] 戻したい時よく使っているコマンドまとめ

<https://qiita.com/rch1223/items/9377446c3d010d91399b>

6. 直前のコミットのやり直し方

git commit を取り消して元に戻す方法、徹底まとめ

<http://www-creators.com/archives/1116>

(6) virtualbox の容量が足りなくなった時

①Dropbox の容量削減

ubuntu で Dropbox を使っていると、保存しているファイルだけではなく、キャッシュファイルが肥大してディスク容量を圧迫してしまうことがあります。ubuntu のアプリケーションである「ディスク使用量アナライザー」を使用し、Dropbox の容量が肥大化している場合は下記の方法で修正することができます。

1. Dropbox フォルダの中の「.dropbox.cache」が肥大化した場合

【ubuntu】Dropbox の容量が膨れ上がる問題

https://clean-copy-of-onenote.hatenablog.com/entry/Dropbox_cache_problem

2. home フォルダの中の「.dropbox」が肥大化した場合

へぼエンジニアノート

<http://note.kahwi.com/2011/11/dropboxsigstoredb.html>

②ディスク容量の追加

1. ディスク容量の拡大

<https://blog.goo.ne.jp/ashm314/e/e716cb8c4652b99af866a456cd899e89>

2. 拡大した後のパーティションの調整

VirtualBox 環境での Ubuntu の HDD 容量変更方法

https://qiita.com/ryokato_me/items/3b2298f9016a8a002ecd

(7) 指数表記の早見表

<https://qiita.com/rev84/items/68f9c5bf7aa7e8e8e13f>

指数表記	日本語	バイト	備考
e- 9	十億分の一	ナノ	
e- 8			
e- 7			
e- 6	百万分の一	マイクロ	
e- 5			
e- 4			
e- 3	千分の一	ミリ	
e- 2			
e- 1			
e+ 0	一		
e+ 1	十		
e+ 2	百	MySQL:tinyint(2.6e+2)	
e+ 3	千	K(キロ) 競馬:16 頭の三連単パターン数(3.4e+3)	
e+ 4	万	MySQL:smallint(6.6e+4)	
e+ 5	十万		
e+ 6	百万	M(メガ) ミリオン	

(8) RoboCup シミュレーションリーグ秋(冬) キャンプ 2019

合宿

最後にチームのファイルをもらうのを忘れない

①質問事項

- ・ Japancup2020 は rcssserver15.50 (今年と同じサーバー) か？
- ・ 360 度方向に動くパターンの librcsc、agent2d のアップデートはあるか？
- ・ 世界大会の進め方を聞く

robocup2019 の SeedingRound、QualifyingRound、PreliminaryRound、MainRound、Elimination とは何か？

- ・世界大会本戦では、サーバーから相手名が取れるのか
- ・世界大会の `start` ファイルは `JapanOpen` と同じで良いか、フォルダの中にある `start.sh` は使わないか？

②プログラムのアップ

(参考)

この部分は既にセッティング済みですが参考のために記載します。

`openssh-server` のインストール

```
sudo apt-get install openssh-server
```

ホスト名の取得

```
hostname
```

LAN 内 IP アドレスの取得

```
hostname -I
```

1. 初日の手続き

(1) 接続

ホスト名と IP アドレスを使って SSH 接続を行う (22 番ポート)

```
jyosen2019
```

```
soccer
```

(2) ファイルの送付

`teraTerm` を使えばドロップアンドドラックでファイルを送付できる。下記のコマンドでファイルを送るパスを取得し、`teraTerm` に入力する

```
mm@mm-VirtualBox:~/Dropbox/soccer/test2$ pwd
```

```
/home/mm/Dropbox/soccer/test2
```

(3) 送付したファイルを `Jyo_sen` フォルダに移動

```
mm@mm-VirtualBox:~/Dropbox/soccer/test2$ ls
```

```
Jyo_sen  mochizuki.tar.gz  test
```

```
mm@mm-VirtualBox:~/Dropbox/soccer/test2$ mv mochizuki.tar.gz Jyo_sen/
```

```
mm@mm-VirtualBox:~/Dropbox/soccer/test2$ ls
```

```
Jyo_sen  test
```

(4) ファイルの解凍

```
mm@mm-VirtualBox:~/Dropbox/soccer/test2$ cd Jyo_sen/  
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ tar -xvzf mochizuki.tar.gz
```

```
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ ls  
mochizuki  mochizuki.tar.gz  start  start.sh
```

(5) other フォルダを作成し、最初から入っている start,start.sh,kill.sh と mochizuki.tar.gz を移動させる

```
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ mkdir other  
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ mv start other/  
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ mv start.sh other/  
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ ls  
mochizuki  mochizuki.tar.gz  other  
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ mv mochizuki.tar.gz other/  
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ ls  
mochizuki  other
```

(6) mochizuki フォルダの中に入っているプログラムを一つ上の階層である/Jyo_sen フォルダにコピーする

```
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ cd mochizuki/  
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen/mochizuki$ cp -a ..
```

```
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen/mochizuki$ cd ..  
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ ls  
AUTHORS      INSTALL      Makefile.in  README      bootstrap  
config.h      config.status  m4          src         start.sh  
COPYING      Makefile      NEWS        aclocal.m4  changelog.txt  
config.h.in  configure     mochizuki   stamp-h1    start2.sh  
ChangeLog    Makefile.am   NEWS.en     autom4te.cache  config      config.log  
configure.ac  other        start
```

(7) 不要となった mochizuki フォルダを other フォルダに移動させ、プログラムが実行可能かのチェックを行う


```
mm@mm-VirtualBox:~/Dropbox/soccer/test2/Jyo_sen$ mv mochizuki/ other/
```

(8) パスワードの変更

```
passwd
```

```
robocup2020
```

2. 2 日目の手続き

(1) または (2) の方法で前日のファイルを片付ける

(1) 1 日目のファイルをすべて消す場合

```
rm -r *
```

(2) 1207 というフォルダを作って、前日のファイルをすべてその中に移す場合

```
mkdir 1207
```

mv コマンドを使って順番に移動する方法は分かりますが、

任意のフォルダの中の 1207 ディレクトリを除くすべてのファイルとディレクトリを、
一括で 1207 フォルダに移す方法は分からないので、方法が分かりましたら教えて下さい。

(3) 初日と同じ方法でファイルをアップロードする

3. データのダウンロード

最終日に他のチームのデータをダウンロードする

(1) 圧縮する (退会では他チームのファイルはすでに圧縮されていると思います)

```
tar -cvzf Jyosen2019.tgz jyosen2019
```

(2) ダウンロード

テラタームの「ファイル」→「SCP…」を選択して、ファイルへのパスを入力して実行する。